Thesis for The Degree of Doctor of Philosophy

Cryptographic Tools for Privacy Preservation

Carlo Brunetta

Department of Computer Science & Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2021

Cryptographic Tools for Privacy Preservation

Carlo Brunetta

ISBN 978-91-7905-528-8
Doktorsavhandlingar vid Chalmers tekniska högskola, Ny serie nr 4995.
ISSN 0346-718X

Technical Report No 205D
Department of Computer Science & Engineering
Chalmers University of Technology
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Gothenburg, Sweden 2021.

"Breathe, breathe in the air...
...don't be afraid to care..."

"Breathe" - Pink Floyd

Abstract

Data permeates every aspect of our daily life and it is the backbone of our digitalized society. Smartphones, smartwatches and many more smart devices measure, collect, modify and share data in what is known as the Internet of Things.

Often, these devices don't have enough computation power/storage space thus outsourcing some aspects of the data management to the Cloud. Outsourcing computation/storage to a third party poses natural questions regarding the security and privacy of the shared sensitive data.

Intuitively, Cryptography is a toolset of primitives/protocols of which security properties are formally proven while Privacy typically captures additional social/legislative requirements that relate more to the concept of *"trust"* between people, *"how"* data is used and/or *"who"* has access to data. This thesis separates the concepts by introducing an abstract model that classifies data leaks into different types of breaches. Each class represents a specific requirement/goal related to cryptography, *e.g.* confidentiality or integrity, or related to privacy, *e.g.* liability, sensitive data management and more.

The thesis contains cryptographic tools designed to provide privacy guarantees for different application scenarios. In more details, the thesis:

(a) defines *new encryption schemes that provide formal privacy guarantees* such as theoretical privacy definitions like Differential Privacy (DP), or concrete privacy-oriented applications covered by existing regulations such as the European General Data Protection Regulation (GDPR);

(b) proposes new tools and procedures for providing *verifiable computation's* guarantees in concrete scenarios for post-quantum cryptography or generalisation of signature schemes;

(c) proposes *a methodology for utilising Machine Learning* (ML) for analysing the effective security and privacy of a crypto-tool and, dually, proposes a secure primitive that allows computing specific ML algorithm in a privacy-preserving way;

(d) provides an *alternative protocol for secure communication* between two parties, based on the idea of communicating in a periodically timed fashion.

Keywords

Cryptography, Privacy, Outsourced Computation, Cloud Computing, Verifiability

Acknowledgment

Let me start by thanking my supervisor, **Katerina**. The many conference's rejections, the long research visiting, the pandemic and many other complications. It was definitely tough (for both of us) but it was fun and educational!

Next, I would like to thank my co-supervisor **Bei**. Always prepared and ready to keep the work going and, additionally, an amazing office-mate with whom share mundane discussions regarding food, politics, economics and food (again, yes). I'm looking forward to coming to visit you in Beijing, either for work or for (food) holidays!

It is really hard to write a complete list of names, but I want to deeply thank all the many people in **my division/unit, Network and System**, for sharing the good/bad moments of the daily work. Additionally, thanks to all the **administration "moms and dads"** for all the support that they gave me either *"work-related"* or *"it's a blue day"*. **Tack <3**

A big thank you to the uncountable number of friends that crossed my life here in **Chalmers**. Thank you for all the good Fika, the beers and all the afterworks that made the journey a little more chilled.

A special *"efharisto poli!"* goes to a (crazy) friend and co-worker, **Georgia**. Thank you for all the laughs, drinks and philosophical discussions! It was really nice to share all these crazy years together. I wish you to finish the PhD soon, the best for your future, a lot of luck and to continue travelling the world!

Another special thanks go to **Pablo, Lara, Oliver and Erik**. It is definitely a pleasure seeing an amazing family grow and I really wish you the best of luck for all the future challenges!

An enormous *graxie* goes to **Elena**, my unofficial tutor, guide and co-worker and, mainly, an amazing Friend, with capital *"F"*. You helped me a lot at the beginning of my *crazy journey*. You and your amazing wife **Hedvig** were always there to give good, sincere advice and meaningful help. It is definitely hard to explain our (Aura's and mine) gratitude for how amazing you two are and I will not even try. I prefer to promise that we will continue sharing our path, share the good and bad moments and try to get together and having a good meal, a couple of drinks and enjoy every moment together, anywhere on Earth. (==)

Outside work, I was incredibly lucky to have found a multitude of incredibly amazing Friends, again with capital *"F"*. We shared different hobbies, interests, opinions but, most of all, we shared many unforgettable, meaningful and once-in-a-lifetime moments. Thank you to all my Sahlgrenska's real-science friends **Tugce, Lydia, Eleni, Axel, Giacomo, Alina, Masako and many more** and the *"climbing monkeys"* **Jasmine, Eridan, Clement and Katja**. Of course, I cannot forget to thank new friends like **Martin, Simon, Johan, Isabel, Veronica**, the old ones like **Alberto "Benjo", Davide, Kevin, Kekko, Andrea, Giorgia, Mia, Marta, Gloria, Silvia, Fede, Gloria, Alice, Mattia, Dylan, Seba, Casa, Costa, Tommy, J, Anto, Maru, Fox and many, many, many others** and, of course, the Vichy's crew with **Captain Moch, Sbrilli, Anton, Marta, Eros, Fede and Mamma Ambra**. Furthermore, a

special thanks go to the *"Band with Many Names"* composed by **Marco, Enzo, Pier, Evgenii and Grischa**. We shared a lot of adventures and I'm really grateful for all the good bohemian moments and improvised jams. Our band will always be a beautiful memory in my musical career.

Thank you, my Friends, for every moment. It is indeed the $\mathbb{R}$-life and I know that our path might diverge. Already many of us are getting married, having our first baby and/or moving to other countries. Our lives are slowly turning to different paths and I feel a little sad about it. But if I feel sad, it means that I cared a lot about our Friendship thus I wish you all the best for your career, family, happiness and any type of goal. We will definitely meet again, one day, and just *"synchronize"* our new experiences, adventures and achievements!

Continuing the emotional side of the section, I want to thanks Aura's family: **Paolo, Manuela and Alice**. Grazie per avermi accolto nella vostra famiglia e per tutto il sostegno e l'aiuto che date a me e ad Aura. Grazie mille per tutto!

I will switch to my dialect to properly thank my family, **Bepi, Reza and Gigi**.

Prima de tut, Graxie, con la *"G"* granda. So de esar al fiol pì casinaro e so benisimo che no le stat facile vedarme volar via all'estero, cresar così velocemente quasi da no riconosérme pì. Ma savee benisimo che se son così bravo in tel me laoro, rispetà e amà da tuti i me Amighi, a le solo graxie a come che me avé cresest. Graxie par averme soportà, par averve *"cavà al pan dala boca"* par darme an futuro diverso, milior de quel che avé pasa voi. So che mi e Gigi sion i vostri punti de orgoglio. Dove saver anca voi che son sempre fiero, orgoglioso e content de chi che sie, dei vosti enormi sacrifici e dela enorme umiltà che ve rende così unici. Se son quel che son, a le solo graxie a voi. E ora che la pension se avicina, vede de goderve al vostro meritato riposo. Graxie de tut.

As you might expect, I left the best for the end!

Thank you, my love. Really really thank you, **Aura**. You fill my days of love, energy and (good) no-sense, you give me reasons for fighting for a better Universe, you definitely make me a better human being (a quite awkward but still a better one!). I love you and you know it. No infinite amount of ink can describe how important you are. We share the best moments of our lives and I'm so eager to see where our future will bring us. All the new adventures, achievements and challenges.

You always call me your *"Mountain"* because I sometimes make you mad when I'm too introverted, cold and harsh. But, on the positive side, I'm there, stable and calm, ready to shift the whole Universe only to see you happy. Coming from the Alps and by looking at my personality and hobbies, I definitely feel like a Mountain.

You are my precious *Stella Alpina*.
You are part of me and you make me important, you make me proud of who I am, you make me want to protect you from all the tourists that are trying to pick you up and that doesn't know how strong you really are[1]. I believe that the *"Sea"* better represents who you are. Peaceful but incredibly strong, deep but calm on the surface. We complete each other, I'm the Mountain, you are the Sea.

I'm not a good swimmer (as we can agree from this last holiday) but there is something that I love doing: I love to look at the horizon, being from the top of a mountain or the shoreside. It makes me think of the past, the present and the future. It brings me peace, as you do, every day. And whenever you bring me peace, I'm able to appreciate all the love you give me and, to me, *our love is all that matters.*

[1] Stelle alpine are astonishingly strong and brave! Like, deciding to live in between harsh rocky terrain, under the freezing winter snow and strong winds only to pop out in the late summer, to enjoy the sun and the immense silence and peace that only the highest mountains can provide. That's hardcore!

I may have forgotten amazing people that crossed my work and personal life. I'm technically writing this section during my holidays, so sorry about my bad memory! If you are not on this list, don't feel angry. Just let me know and I will offer you a drink!

I concluded my licentiate acknowledgement with a quote from the masterpiece *"Dark Side of the Moon"* and I admit that it is a perfect summary even now:

> For long you live and high you fly,
> and smile you'll give and tears you'll cry,
> and all you touch and all you see,
> is all your life will ever be.
>
> *Breathe* - Pink Floyd

Appended Publications

This thesis is based on the following publications:

Paper A: **C. Brunetta**, C. Dimitrakakis, B. Liang, A. Mitrokotsa
"A Differentially Private Encryption Scheme"
20-th Information Security Conference (ISC), 2017, Ho Chi Minh city (Viet Nam). Spinger, LNCS, Vol. 11124, 2017, pg. 309326. [BDLM17]

Paper B: E. Pagnin, **C. Brunetta**, P. Picazo-Sanchez
"HIKE: Walking the Privacy Trail"
17th International Conference on Cryptology And Network Security (CANS), 2018, Naples (Italy). Springer, LNCS, Vol. 10599, 2018, pg. 4366 [PBP18]

Paper C: **C. Brunetta**, B. Liang, A. Mitrokotsa
"Lattice-Based Simulatable VRFs: Challenges and Future Directions"
1st Workshop in the 12th International Conference on Provable Security (PROVSEC), 2018, Jeju (Rep. of Korea) and *Journal of Internet Services and Information Security, Vol. 8, No. 4 (November, 2018).* [BLM18]

Paper D: **C. Brunetta**, B. Liang, A. Mitrokotsa
"Code-Based Zero Knowledge PRF Arguments"
22-th Information Security Conference (ISC), 2019, New York (USA). Spinger, LNCS, Vol. 11723, 2019, pg. 171-189. [BLM19]

Paper E: **C. Brunetta**, B. Liang, A. Mitrokotsa
"Towards Stronger Functional Signatures"
Manuscript.

Paper F: **C. Brunetta**, P. Picazo-Sanchez
"Modelling Cryptographic Distinguishers Using Machine Learning"
Journal of Cryptographic Engineering (July 2021), [BP21].

Paper G: **C. Brunetta**, G. Tsaloli, B. Liang, G. Banegas, A. Mitrokotsa
"Non-Interactive, Secure Verifiable Aggregation for Decentralized, Privacy-Preserving Learning"
To appear in 26th Australasian Conference on Information Security and Privacy (ACISP), 2021, Perth (Australia).

Paper H: **C. Brunetta**, M. Larangeira, B. Liang, A. Mitrokotsa, K. Tanaka
"Turn Based Communication Channel"
Manuscript under submission.

Other publications

The following publications were published during my PhD studies, or are currently under submission. However, they are not appended to this thesis.

(a) **C. Brunetta**, M. Calderini, and M. Sala
"On hidden sums compatible with a given block cipher diffusion layer"
Discrete Mathematics (Journal), Vol. 342 Issue 2, 2018 [BCS19]

(b) G. Tsaloli, B. Liang, **C. Brunetta**, G. Banegas, A. Mitrokotsa
"DEVA: Decentralized, Verifiable Secure Aggregation for Privacy-Preserving Learning"
Manuscript under submission.

Research Contributions

Paper A: I was involved in the initial brainstorming with Aikaterini and Christos who proposed me the idea of including differential privacy in the cryptographic domain. I had the idea of relaxing the correctness property of an encryption scheme, the key idea that allows defining differentially private encryption schemes. I further formalized, defined and proved all the contents of the paper. In the final stage, I wrote the implementation and the statistical tests.

Paper B: after many fruitful morning-fika and brainstorming with Elena and Pablo (and Oliver!), we all together traced the main structure and motivation for the HIKE protocol. During the development of the paper, I was the relay figure for the translation between theory and implementation. More specifically, I wrote the draft of some proofs and I was responsible for the theoretical aspects necessary for the implementation. Finally, I am the corresponding author of this work and I finalised the camera-ready version.

Paper C: I participated in the initial brainstorming discussion with Bei and Aikaterini after Bei's suggestion on the specific topic of constructing a post-quantum verifiable Pseudo-Random Function. I completely wrote the first draft of the paper. After receiving some useful external feedback on the paper, I participated in finding different possible solutions while Bei and Aikaterini revised the draft. In this final and much shorter version, I conceived the summary of the entire research-exploration and I was responsible for the introduction-background sections of the final paper.

Paper D: Bei, Aikaterini and I jointly discussed the possibility of extending Paper C's methodology for code-based cryptographic assumptions. I discovered and developed the content of the paper, wrote proofs and I completely wrote the first draft of the paper. After receiving some useful feedback on the paper from Bei and Aikaterini, I finalised the paper.

Paper E: I participated in the initial brainstorming discussion with Bei and Aikaterini after Bei's suggestion on the specific topic of providing construction for extending the Functional Signature primitive with a verifiability property. I was responsible for designing the Strong Functional Signature (SFS) instantiation with the related security proofs. In the first draft, I wrote the instantiation, security proofs and general introduction. After receiving some useful external feedback on the paper, I took the responsibility of revising the SFS's primitive, security model/properties and the application described in the introduction.

Paper F: after several discussion with Pablo, we together traced the main structure and motivation for a methodology for generating cryptographic distinguishers using machine learning. I was leading the project and developing the theoretical framework. I designed and performed the statistical analysis of our framework's experiment. I wrote the majority of the first draft and handled the journal communications.

Paper G: I joined the discussion with Georgia, Bei, Aikaterini and Gustavo regarding distributed federated learning. Concurrently, I designed a non-interactive primitive while Georgia and Bei defined DEVA (Paper b). After receiving some useful external feedback on the first paper, we jointly decided to split the constructions into two papers (Paper G and b) and I took the responsibility for my construction's paper. Thus, I defined and proved the security of NIVA, I wrote the first paper draft and I helped to debug minor problems in the implementation.

Paper H: I had the original idea of developing a turned communication channel which I later developed initially with Aikaterini and Bei and later with Mario and Keisuke during a research visit. I led the project and, in the first paper version, I wrote the initial draft of the protocol's construction and introduction's section. I double-checked the fairness security and proof that Mario and Keisuke wrote. After receiving some useful external feedback on the paper, we decided to split the paper into concrete instantiation and the theoretical implications of our construction. Currently, Paper H contains the protocol instantiation that I initially wrote.

All the co-authors agree on the preceding statements.

Thesis Contents

Introduction

Every single day
Every word you say
Every game you play
Every night you stay
I'll be watching you

Every Breath You Take - The Police

Our society lives in an era where every device, electronic or not, is becoming *"smart"*. Smartphones, smartwatches, smart glasses are examples of many new devices that are continuously being constructed and introduced in our daily life. All these *smart devices* are designed to improve productivity, automatise tasks and track complex procedures. This is possible by providing the devices with the ability to manage **data** by providing them with *computational power* and the ability to *communicate* with each other.

More precisely, the adjective *"smart"* relates to the device's ability to handle *"data management"* which can be classified into the actions of *(i)* **generating**; *(ii)* **communicating**; *(iii)* **storing**; and *(iv)* **computing/manipulating** data. In other terms, a smart device is a *"standard"* device that incorporates a computer-like microcontroller able to capture the device status, manipulate the information and communicate it to other smart devices.

This simple concept allows the consideration of *hyperconnected networks* of (often low) computational devices, better known as the *Internet of Things* (IoT). The IoT principle is based on the ubiquitous presence of cheap and low-computational devices that constantly generate, collect, manipulate and share data locally between themselves or with a *"higher entity"* called *the Cloud*.

For example, consider the thesis' writer, Carlo, that lives in a *smart home*, *i.e.* a home where lights, smart electro-domestic and more sensors/actuators are interconnected on the same home-local network. All the data collected throughout the house is, often, centrally collected on a house-router that later uploads part of the data to an external service *"on the Cloud"*. Abstractly, the Cloud is an *interface of data management services* that any authorised smart device contacts via the Internet and utilises to *"simplify"* the data processing. Despite the Orwellian feeling of massively collecting data and centralising it into a single external entity, the Cloud provides useful analysis to the router and allows Carlo to better control every measurable aspect of the home.

For example, Carlo might be highly interested in maintaining high-quality air in his home. To do so, Carlo's house is filled with air-quality sensors that collect pollution data, send it to the central router which later *"ask the Cloud"* for an analysis. Since this collecting-analysis is continuously executed, Carlo has the power to check the air-pollution in his house at every moment. This means that Carlo can voice-activate its home-assistant device and ask *"which room has the cleanest air?"*, the device will record Carlo's command and upload the recording to some voice-recognition service *"on the Cloud"* that will transliterate the command's request.

Whenever the home assistant receives the request transcription, it will ask the home-router an answer which will, most probably, *"contact the Cloud"* that will analyse the request and reply to the router with the answer. After all this back and forth, the router will provide the assistant with the answer that can effectively be announced verbally to Carlo after just a couple of seconds.

The careful reader might notice the writer's highlight of actions referred to *"into/to the Cloud"*. The reason for such pedant highlight is the necessity to take a step back and precisely delineate the concrete reality of the Cloud's *"composition"*. Similarly to the atmospheric homonymous and depicted in Fig. 1, the Cloud is a network conglomeration of smaller networks of computers, all interconnected and orchestrated to appear as a *"hyper-computer"*, *i.e.* a computer with incredible computational power, unimaginable storage capacity, extremely efficient communication bandwidth and always available. The quintessential aspect is that *"to use the Cloud"*, the user **does not** need to know where these computers are, their characteristics, how they operate or how they are organised. The writer's highlight wants to point out that *"uploading to the Cloud"* is, fundamentally, semantic sugar for *"uploading to some unknown-but-retrievable computer on the Internet"*.

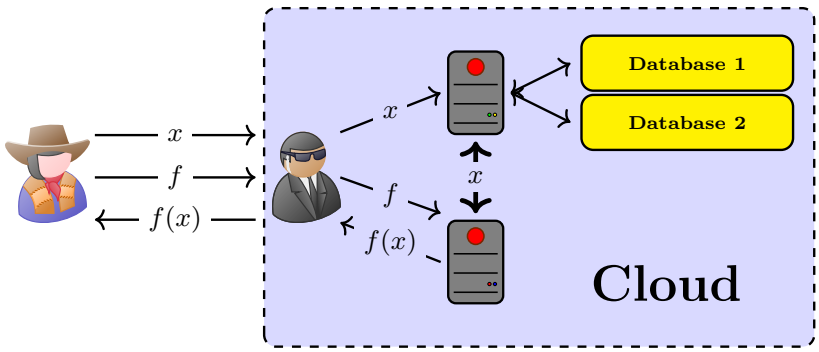


Figure 1: Picturesque representation of the Cloud's composition.

Data is **the** fundamental element of our digital society and imposes a remarkable role on our digital identity. Generated data can either be *public* or *sensitive/private* depending on the *data owner* thus requiring different confidentiality guarantees whenever handled. The IoT paradigm is based on having the smart devices execute part of the data management via cloud computing which, concretely, can be seen as simply requiring the devices to *outsource computation* to a more powerful computer. In other words, all the devices' data is handled by *unknown* computers on the Internet.

How is it possible to ***trust*** *the Cloud to* ***properly handle*** *user's sensitive data?*
What does it mean "to trust someone" *and* "properly handle data"*?*

Throughout history, humans evolved their *secrecy's needs* into the **cryptography** discipline. Figuratively, cryptography is *the toolset* of algorithms and protocols that allows the user to provide confidentiality, integrity, authenticity and many other properties that handle sensitive data. As in any proper toolset, there are several tools from *must-have screwdrivers*, such as the Diffie-Hellman's key-agreement protocol, to multi-purpose *Swiss Army-knifes*, such as the Fully Homomorphic Encryption (FHE) schemes. The main objective for all cryptographic tools is to avoid any data leaks, *i.e.* each one of these tools is designed to provide precise *security guarantees* which are formally defined and mathematically proven, *e.g.* confidentiality, integrity, authentication, anonymity

and many more. The use of formal modelling is fundamental to unequivocally describe how a cryptographic tool must be used to achieve the security guarantees when it can be used and all the limitation that it might have. The usage of mathematics for describing the cryptographic elements allows us to firmly state that a provably secure crypto-tool can not be the cause of a *data leak*, *i.e.* the scenario in which a malicious entity can disrupt/break the provided tool's security guarantees. On the contrary, if an adversary can *"break the crypto-tool"*, then either the cryptographic primitive/protocol or the security model used is not secure thus it is impossible to formally prove the tool's security or model's usefulness.

Often used in daily conversations, a different concept to consider is **privacy**. The main goal for privacy is complex and it is highly related to *how data is used* and *how to prevent data to be harmful* which require an extensive analysis of the application that requests privacy guarantees. Each privacy guarantee is an *"interdimensional"* requirement that spans from cryptographic security requirements to real juridical liability, business' responsibility or human necessities. In a nutshell, the concept of privacy is *"the framework"* that provides real/legal guarantees to people that their data is not misused in a harmful way.

Privacy and cryptography define a *spectrum* of requirements that describes the trade-off between *security and usefulness* and can be associated with the concept of *trust*. On one side of the spectrum, we have the *"no-trust"* scenario where the user's data is required to be secret, where no one else than the data owner can access the data. On the other side, the *"only-trust"* scenario where the same user's data might be communicated unencrypted with the only requirement of *"not misusing this information"*.

Hidden in the scenario's description, the spectrum naturally introduces the concept of *shared data* between users, *i.e.* someone else's private data which shouldn't be misused. Any privacy guarantees require shared data to be protected because it requires the data owner to trust the receiver not to misuse such sensitive information. At a first glance, protecting shared data might appear as a different way to name private/secret data but it is essential to understand that it is possible to lose all the privacy guarantees without breaking any used cryptographic tool. Consider a user that securely uploads to the Cloud a private photo of him/her and let the user fully trust the Cloud to maintain the necessary secrecy. Despite the cryptographic guarantees that the communication is secure, the photo is most probably unencrypted for the Cloud which utilises the photo for improving its services, *e.g.* trains classifiers for better face recognition. Without breaking any crypto tool, the Cloud can break the user's trust and publicly release the private photo thus breaking the trust agreement between itself and the user.

This discrepancy between cryptographic and privacy requirements is described in several legal regulations such as the California Consumer Privacy Act (CCPA) of 2018 [Par18] or the European General Data Protection Regulation (GDPR) [Cou16]. These regulations, and many more, provide a *legal foundation* that precisely state which user's data is *sensitive* thus requiring the Cloud's special care while handling the data. The regulations further describe precise liability penalties whenever a user's data is misused. For example, the user's IP address is sensitive information that can be maliciously used to approximately geo-localise the user or track him/her throughout the web. It is fundamentally impossible to navigate the web without revealing the personal IP address thus the servers must correctly handle this, and other, sensitive data. Otherwise, the users can bring the server's owner to court for misusing sensitive data.

To understand the differences between cryptographic and privacy guarantees and further provide future research directions in the intersection area of cryptography and privacy, it is mandatory to provide an abstract analysis of all the possible data leakage that might occur between any interaction of two entities.

1 Abstract Model for Data Leaks

People own collections of personal data and each one of them partitions the collection based on the specific data's **sensitivity**. More formally each person P_A classifies data into the collections of:

- **private data** $\mathfrak{C}$ that contains any information that P_A is **not willing to share** with anyone else. These are highly sensitive data that a malicious entity can use to seriously harm P_A thus must be carefully handled;
- **shared data** $\mathfrak{S}$ that contains P_A's private data that is **consensually shared** with a different person P_B. Because such data is technically private, P_A must trust P_B to not misuse/publish the shared data. On the other hand, P_B uses the data to provide some form of benefit to P_A, *e.g.* a personalised service. This data collection is strictly connected to trust and the concept of privacy;
- **public data** $\mathfrak{P}$ that contains P_A's public data that is **freely shared** with anyone. Ownership of such data cannot be used to harm P_A and are therefore easily retrievable.

For example, Carlo considers the data x = "work email address" to be public while ξ = "personal email address" is more sensitive so it is only shared with selected other people and web services. Consider the last example where Carlo considers ξ = "personal email address" $\in \mathfrak{S}$ and uses ξ to register to a generic social network $\mathcal{N}$. A (quite typical) scenario is that the social network $\mathcal{N}$ will publicly display ξ by default because $\mathcal{N}$ considers $\xi \in \mathfrak{P}$. This notion is condensed into the following axiom:

Informal Axiom 1. *Data partitioning is* ***subjective,*** i.e. *every person P has his/her way of partition data into* $(\mathfrak{C}_P, \mathfrak{S}_P, \mathfrak{P}_P)$.

Sadly, Informal Axiom 1 implies that deciding the sensitivity of a specific data is **ill-defined**, *i.e.* it is not possible to uniquely identify the correct partition to which data belongs, as previously described.

Additionally, data appears to be *"naturally entangled"* with other data, as if it is semantically interconnected. Intuitively, from big sets of information, it is possible to **infer** new information, maybe without absolute certainty thus requiring some probabilistic discussion. For example, if Carlo would present itself with a wet umbrella, the reader can deduce that it is raining outside. Or, by observing Carlo's smartphone screen, the reader can infer his usage pattern by analysing the *"oily"* residues left on the screen. Furthermore, Sherlock Holmes might be able to deduce the pin-code digits' used to unlock the phone by analysing the shape of the oily fingerprints. By carefully reading the examples, observe that Carlo might be unaware of how his data can be maliciously used when combined with *"advanced detective's knowledge"*.

Informal Axiom 2. *Data is* ***always dependent*** *on other data: for every information z, there always exists a set $\{x_i\}_{i\in I}$ that infers about z,* i.e. $\{x_i\}_{i\in I} \to z$.

Informal Axiom 2 describes two negative corollaries which state, from some known information x, the impossibility to compute *(i)* all the *inferable* data z, *i.e.* all the z such that $x \to z$; and *(ii)* all the data-sets $\{z_i\}_{i\in I}$ that infers about x, *i.e.* $\{z_i\}_{i\in I} \to x$.

The axioms allow the analysis of all the possible inference between the different sensitivity partitions, *e.g.* the inferences that take private data $\{s_i\}_{i\in I} \subseteq \mathfrak{C}$ and infers a public information $y \in \mathfrak{P}$. By conceptually reasoning on the empirical meaning of such deductions, the final result is an abstract model that describes a classification of any

data leak into four semantically different breaches, represented in Fig. 2 and named: *(i)* security breach; *(ii)* direct breach; *(iii)* coercion breach; *(iv)* indirect breach.

Before moving to a precise analysis of each breach, it is important to remark on an indirect consequence of Informal Axiom 1. As in any good model, the data leak classification into breaches is *relative to the observer*, *i.e.* the leak might hurt P_A but benefit P_B and it is caused by their different data sensitivity partitioning.

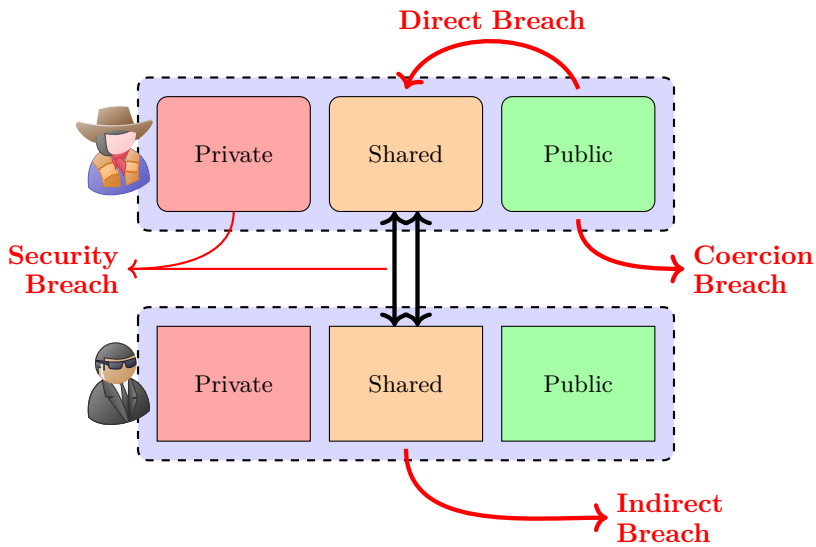


Figure 2: Data leak's model from the cowgirl's point of view. The black arrows indicate the communication between the parties. The red arrows indicate all the possible data leaks.

1.1 Security Breach

Security breaches are defined whenever an adversary $\mathcal{A}$ can *"break"* the cryptographic primitives/protocols used and the security properties requested, *e.g.* $\mathcal{A}$ decrypts an encrypted database of private data or can compromise the integrity of a secure communication channel.

A historical and didactical example is the cryptanalysis advances that, during the Second World War, allowed the Allied powers to break the encrypting machine *Enigma* used by the Axis powers. Preceding and motivating the development of the first computers, Enigma is an electro-mechanical encrypting device that appears to have a physical typewriter-like keyboard and display of light-emitting characters representing the keyboard. To encrypt, the operator presses a single character key which closes an internal electrical circuit that lights up a precise character in the display. Internally, the machine is composed of rotors that rotate at every typed character, modifying the circuit and the highlighted encrypted output, as represented in Fig. 3. The security of the device is due to the immense amount of possible starting combinations of the rotors and other external additional modifications of the circuit made via a plugboard. Enigma was considered *unbreakable*.

During the war, the Allied power developed the theoretical foundations of Inform-

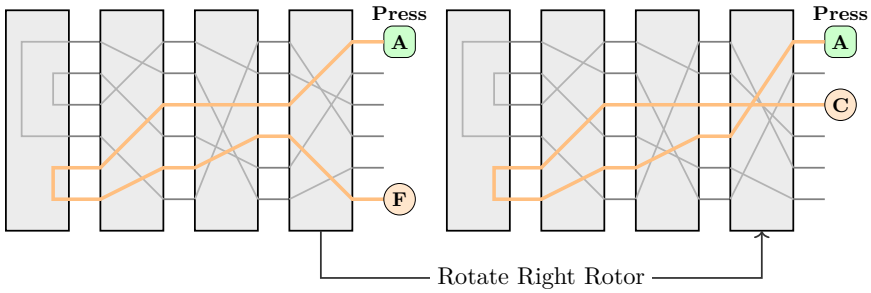


Figure 3: Conceptual illustration of the Enigma machine's encryption principle.

ation Theory [Sha48] and Cryptanalysis. Briefly speaking, together with practical examples of correct decryption, code-books and capturing some Enigma machines, this new knowledge allowed a refinement on the brute-forced decryption attacks which allowed to decrypt the secret communication and provide useful intelligence on the field. In other words, Enigma was *broken*.

In the same spirit, security breaches happen because either the cryptographical knowledge evolves and new successful attacks are being developed or, more simply, the wrong crypto-tool is used. The state-of-the-art primitives/protocols are secure up until the hypothesis used to formally prove the tools' security guarantees holds. This requires researchers to constantly check that new attacks don't break such hypothesis and promptly report to the community whenever a crypto-tool is broken.

1.2 Direct Breach

Direct breaches are defined whenever it is possible to deduce private/shared data from public ones. Despite the simple definition, these breaches are intrinsically sneaky to identify and prevent.

In October 2006, the on-demand streaming service Netflix released a dataset containing hundreds of millions of *private* movie ratings generated by half a million subscribers. The release's purpose was to allow the development of an improved movie recommendation system. To guarantee privacy, the dataset was *anonymised*, *i.e.* the subscriber's sensitive data such as user id, email addresses and even the timestamp of the rating submission was removed. In principle, *only public data was released*.

A couple of years later, Narayanan and Shmatikov [NS08] were able to de-anonymise the identity of known subscribers from Netflix's dataset and obtain his/her movie ratings, thus discovering unexpected sensitive information such as political preferences. Such a surprising result was possible by considering *additional information* such as the one retrievable by personally asking naive questions like *"what do you think about this movie genre?"* or, more systematically, utilise the *public* movie ratings provided by the Internet Movie DataBase (IMDB). The reader might argue that *"de-anonymising movie ratings don't sound harmful"* but consider the scenario where a malicious adversary $\mathcal{A}$ can de-anonymise the identity of the ratings. Only because $\mathcal{A}$ can de-anonymise people from their *"movie tastes"*, $\mathcal{A}$ can *profile* the unlucky subscriber and increase the ability to track him/her throughout the Internet.

Direct breaches are caused by the Informal Axiom 2 and the impossibility to conceive all the possible deductions that public information can provide. Conceptually, note that it is not obvious *how* cryptographic tools can protect from such breaches. For

this reason, the state-of-the-art solution is found in the concept of **Differential Privacy** [DMNS06] (DP) which provides a formal framework to measure the privacy loss of publishing specific data related to a dataset. To understand how DP works, consider a private dataset of values $\{x_i\}_{i=1}^n$ on which it is required to compute the known function f. The computed output $\mu = f(x_1, \cdots, x_n)$ is publicly released thus meaning that $\{x_i\}_{i=1}^n \to \mu$. Without loss of generality, by cleverly modifying the function's input, it might be possible to obtain the public value $\mu' = f(x_2, \cdots, x_n)$ in which the private data x_1 is **not used**. The direct breach, as represented in Fig. 4, is caused by considering the function f and the public outputs μ, μ' and observing that any *difference* between outputs must relate with x_1, *i.e.* the breach tries to deduce $\{\mu, \mu', f\} \to x_1$.

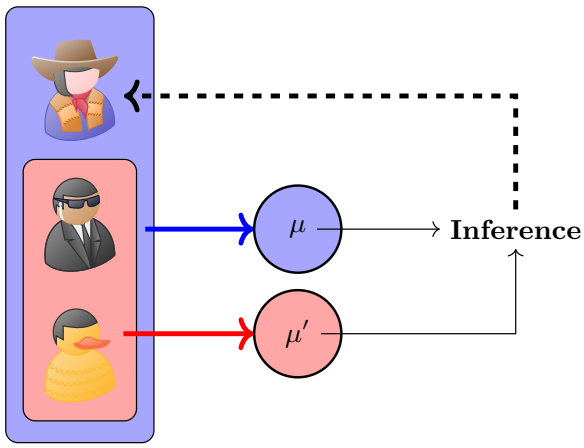


Figure 4: Depiction of the problem solved by the differential privacy framework.

DP provides a methodology to *measure* the privacy loss caused by releasing f's outputs and, to avoid the breach, a DP mechanism adds **noise** which is sampled by a cleverly selected distribution based on the previous measurements. The key concept of *adding cleverly selected noise* might sound counterproductive but finds roots in the idea of *"degrading the information accuracy"*. For example, by publishing Carlo's birth season instead of the month, the probability of guessing his birthdate is degraded thus a loss inaccuracy.

1.3 Coercion Breach

To understand what coercion breaches are, consider public information x related to some private data of the person P_A. Since x is public, a malicious adversary $\mathcal{A}$ might voluntarily advertise a *false*-statement x' that hurts P_A's image/reputation. The *"coercion"* adjective appears whenever considering that, to clarify that x' is false and x is true, P_A must provide private data y so to allow the inference $y \to x$ thus the adversarial coercion.

A real example of such malicious persuasion can be found in the widespread phenomenon of **media distortion** in which fake news are most probably the easier attack vector. Without entering the immense domain of human psychology, it is well-known that people can easily be influenced by only providing modified photos or provide emotionally intense messages. These cheap modifications are repeatedly shown to allow people to unconsciously change their mind regarding, *e.g.* political beliefs [AG17] or

memories of well-known historical events [SAL07]. The social damaging impact of media distortion through fake news is massive and must be prevented.

Coercion breaches are an undesired consequence of Informal Axiom 2 and the fact that often private data is necessary to understand how public data is deduced. Avoiding these breaches is a *tricky problem* that requires taking into consideration the social aspects of human psychology and it seems counter-intuitive that a cryptographic tool might help.

A possible solution would require appropriate experts to educate people on *digital etiquette* and *critical thinking*, *e.g.* by teaching the importance of source verification and awareness of possible media distortion practices. Observe that the appropriate usage of crypto-tools can help to discover data misuse by providing specific security guarantees or, naively, people might be aware of the *meaning* of the tools guarantees.

1.4 Indirect Breach

The last class in our model are the indirect breaches which are a negative consequence of sharing private data x to some other person P which is trusted to not misuse x. Whenever P misuses x, the assumed trust is lost and there is a data leak and the indirect breach. Whenever reading, in our daily life, news about data leaks and related privacy loss, often the news describes an indirect breach.

Purely for explanatory reasons, consider a *run tracking application*, *i.e.* web application that allows users to collect data, such as their heartbeat, pace and much more, from their running activities with the benefit of providing statistics, professional training advices and more user control on their activities. One such application is Strava [Str18] which allows users to provide precise geo-localisation data, *i.e.* GPS-data. Later on, the users visualise the GPS-data on a map thus allowing each user to correlate, *e.g.*, their pace with the topological morphology of the terrain. Strava, like all the others, is often trusted by its users to securely handle the sensitive data, *e.g.* GPS-data is commonly accepted and shown to be incredibly sensitive data [SSM14].

Having a lot of data allows providing interesting features to the users. One of them is Strava's *"popular routes"* which collects the users' GPS-data, finds highly popular routes and provides a popularity list where users now can find each other and share a training session. The feature has the noble motivation of creating a healthy community and increasing the social interaction between the users.

At the beginning of 2018, the noble feature showcased as a popular route a too-regularly shaped one in a scarcely populated, almost desertic, part of Afghanistan. By carefully checking the satellite image of the route, it was possible to discover a *secret* military base [Her28]. An unaware American soldier was periodically training inside the military base, running around an aircraft's runway thus creating a regularly shaped route. Strava's popular route algorithm worked as intended: the soldier was one of the few people in the whole area using the app which implied that his periodically tracked route was the most popular. The indirect breach, consequent trust-loss and legal cost for the data leak's harmful potential were caused by the soldiers' unawareness of Strava's feature **and** Strava's misjudgement on the sensitivity of using the soldier's GPS-data.

In general terms, it is easy to see that indirect breaches are caused by Informal Axiom 1 and the fact that different people have a different opinion regarding data sensitivity. Trust is a difficult concept to generally formalise thus, to avoid such costly damages, many state-of-the-art cryptographic protocols provide some specific privacy guarantees that allow preventing the leak.

A noticeable mention, of a whole research field that tries to avoid indirect breaches, is the research in *Information Flow Control* (IFC). IFC is based on the simple principle that whenever computing an algorithm on data, the algorithm must not be able to output private data given in input, depicted in Fig. 5. In other words, whenever the input is private, specific computational operations are *"prohibited"* because they might be reverted to get the input. By studying the *"allowed"* operations, it is possible to check which algorithms are immune to indirect breaches and are therefore safely executable.

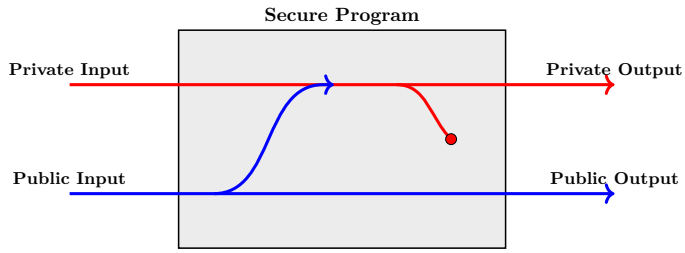


Figure 5: Conceptual representation of the Information Flow Control principle: a secure program does not manipulate the private input and reveals it into the public output.

> Gentlemen. Your communication lines are vulnerable, your fire exits need to be monitored, your rent-a-cops are a tad under-trained...
> Outside of that, everything seems to be just fine. You'll be getting our full report and analysis in a few days, but first, who's got my check?
>
> *Sneakers (1992)* - Martin Bishop (Robert Redford)

As previously stated, it is the research community goal to provide solutions that allow to *"trust the Cloud"* or to avoid any possible data leaks.

The **quintessential research goal** for **any** cryptographic solution that handles people's data is to **avoid data leakages, of any form**.

In other words, *ideal* cryptographic privacy-preserving tools must guarantee *(1)* a tamper-proof data generation; *(2)* secure data communication; *(3)* confidential and privacy-oriented data storage; and *(4)* data computation with *measurable* privacy guarantees, *i.e.* the computed outputs must not reveal *"too much"*.

A key concept that allows reducing the gap between ideal and real solutions is **verifiability**, *i.e.* the property of providing a tangible value used as *"proof"* of either the knowledge of specific information or *certification* of approval. Many existing cryptographic tools already provide verifiability-like guarantees such as:

- signature schemes allow a signer to attach a *signature* to the outgoing messages which can be seen as proof that *"the signer notarises the message content"*. The message-signature pair verification strictly relates to some form of liability that the signer obtains in the act of signing;
- authenticated communication channels, *e.g.* TLS, allow the communicating parties to securely communicate **and** provide the guarantees that only the intended/authorised parties participate in the communication. This is possible by the combination of several different cryptographic tools that are singularly correct and verifiable and that guarantees the confidentiality of the communication and the authenticity of the parties identities;
- in applications, zero-knowledge proofs allow a prover to prove a public statement without revealing the knowledge of a secret witness that easily proves the statement. Being able to provide such verification has profound application scenarios connected to privacy, liability, anonymity and more.

All the described examples provide verifiability for what the user sees or knows and can easily provide verifiability guarantees to data generation, storage and communication. *"Securing data computation"* and providing *"measurable privacy guarantees"* are the missing requirements to tackle.

Data manipulation transforms potentially sensitive information into *new* data that might get published thus having the potential of creating privacy concerns. Quantifying the privacy loss from publishing a computational output is generally hard to compute and/or to correctly and practically handle. For this reason and by observing the problem from a different perspective, it is easier to request **proofs of correct computation** on the data and **control which computation is performed**. It is trivial to see that providing a refined control on the computable functions allows to bound the complexity of computing the privacy loss. Indeed, a trade-off between functionalities and privacy must be considered whenever effectively implementing the system.

Verifying the correct computation of a function allows the verifier to check that the results are indeed correct **and** the correct function was computed. In other words, if something went wrong and the verification fails, the verifier can *identify* the problem, *e.g.* the verifier can precisely shift the data-misuse liability to some entity that later must defend against accusations in the court and not in the cryptographic domain.

To guarantee any form of privacy, it is fundamental to identify any data misuse which is only possible if *every step of the data management is verified.* Ideally, providing (formally provable) verification to every cryptographic tool allows to prevent any data leak:

- any direct breach is caused by a careless release of outputs which allows inferring sensitive data. Requiring the verifiability of the output computation **does not** directly avoid such privacy loss **but** it limits the available computable functions, thus limiting the possible malicious inferences, *and* completely shifts the liability to the publisher. In a sense, these data leaks are solved with the mantra: *"Be aware of what they publish"*;

- verifiability completely solves any coercion breaches since it allows to correctly pinpoint the trustworthiness of the provided data. It must be said that it is always important to **provide a proof** for the computed results and, respectively, to always **request proofs** of the content authenticity;

- security breaches are directly related to the formal security properties that the cryptographic primitives/protocols should achieve. Technically, verifiability is often an additional security property with a really specific description. In other words, the motto is *"always use proven secure and verifiable cryptographic tools"*;

- indirect breaches are always caused by breaking the data owner's trust. Verifiability can prevent these breaches whenever privacy is considered such as **design principles** for new cryptographic tools by providing certainty that the tools are correctly used.

The reality is that to avoid unexpected data leaks, cryptographic tools must be correctly implemented and used as theoretically intended, *i.e.* the purpose they are designed for. The *purpose* is important: there might exist a cryptographic tool that is considered highly secure by the research community, but it is not designed for privacy-oriented applications.

This thesis' goal is to investigate and design new cryptographic primitives/protocols that consider privacy as a fundamental design requirement. By increasing the crypto-toolset with new privacy-preserving crypto-tools, it is possible to choose the appropriate primitive/protocol for real applications thus guaranteeing privacy and security for everyone.

2 Thesis Contributions

This thesis considers several privacy-oriented problems and proposes solutions that formally provide security and privacy-preservation guarantees.

2.1 Differential Privacy and Cryptography

A fundamental principle in Cryptography is that an encryption scheme has to be *correct* and *confidential*, *i.e.* the ciphertext's decryption **must** be the original message and the message cannot be inferred by the ciphertext. Differently, a differentially private (DP) mechanism allows data to maintain privacy when revealed and this is done by introducing a cleverly sampled random noise. Observe that a DP mechanism does not require any confidentiality requirement. This observation brings up the question of combining the two feature:

Question A: A Differentially Private Encryption Scheme

Is there a way to define/construct a differentially private encryption scheme that guarantees confidentiality while data is encrypted ***and*** *afterwards provides a measurable privacy guarantee?*

Paper A consider an encryption scheme and a DP mechanism as a framework and it studies the relation between them to merge them into a single cryptographic primitive.

Contribution: we *relax* the encryption scheme's correctness property. Intuitively, the encryption scheme has to *"wrongly decrypt"* with some bounded and predefined probability, *i.e.* the ciphertext's decryption can return a wrong message m' with some probability $\alpha_{\mathsf{m},\mathsf{m}'}$ that depends on the original message m and the final wrong message m'. The knowledge of such probabilities allows us to prove that the *"faulty"* encryption scheme indeed achieves differential privacy. Additionally, an implementation is provided as a proof-of-concept.

To complete the study, we prove that using such *"faulty"* encryption schemes is equivalent to sequentially using a correct encryption scheme **and** a DP mechanism as two separate frameworks, as depicted in Fig. 6.

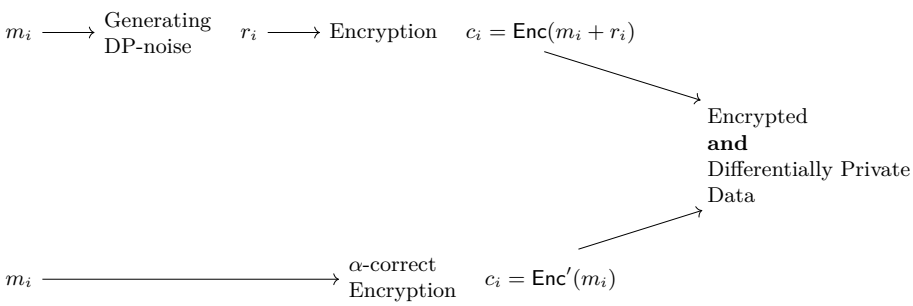


Figure 6: Paper A: The difference between the DP-then-Encrypt (on the top) and our solution (at the bottom).

This means that if we want to introduce differential privacy to already existing products/protocols, it is not required to change the already existing cryptographic

primitives but it is only necessary to introduce a DP mechanism in the system and correctly compose it with the encryption scheme.

2.2 Real Privacy Guarantees by Design

The main goal of Paper B is to provide a model/scheme with an implementation designed to provide privacy guarantees concerning privacy policies/regulations, such as the GDPR, that are not always described in mathematical formalism. By considering the scenario of a user uploading data to a trusted database that can be queried by third parties, the paper answers the following question:

Question B: HIKE: Walking the Privacy Trail

Is it possible to design privacy-preserving protocols that comply with some privacy policies, such as the European GDPR?

We start by selecting some specific articles contained in the GDPR and describe them as formal cryptographic properties:

(a) data has to be encrypted when stored;

(b) the user decides to selectively allow third parties to access his/her data; and

(c) the user can always delete his/her data from the database (*right to be forgotten*).

Contribution: to describe the *"client, cloud and service provider"* model, we use the concept of a *labelled encryption scheme* [BCF17] in which every message, or ciphertext, has a *label* that can be seen as a unique public identifier for that message. With these labels and the associativity and commutativity of the underlying group, we can define *decryption tokens* that can be generated by the client. This allows the user to create decryption tokens for specific label-ciphertexts and provide them to a service provider.

We exploit the additive homomorphic property of the encryption scheme to allow homomorphic evaluations on the client's ciphertexts. In this context, the client can generate decryption tokens for *labelled-programs*, *i.e.* the token necessary to decrypt a specific homomorphic evaluation and defined by the list of inputs, related labels and function to be computed. Since the function must be known to produce the decryption token, the clients can refuse to provide the token and therefore **not disclose** their data.

More concretely, we start from the ElGamal encryption scheme [ElG85], we describe the scheme as a labelled encryption scheme called LEEG, expand it with some specific features regarding the decryption token into FEET and finally obtaining the HIKE protocol, depicted in Fig. 7, that is then proven secure in the GDPR-oriented security model we defined.

As a final contribution, all our ideas are implemented and our code for the HIKE protocol is publicly available.

2.3 Post-Quantum Verifiable Pseudorandomness

Quantum computers are the currently accepted future of computation. Despite the engineering challenges of constructing such a revolutionary machine, the cryptographic research community is interested in providing new primitives that are guaranteed to be secure even against adversaries that use a quantum computer.

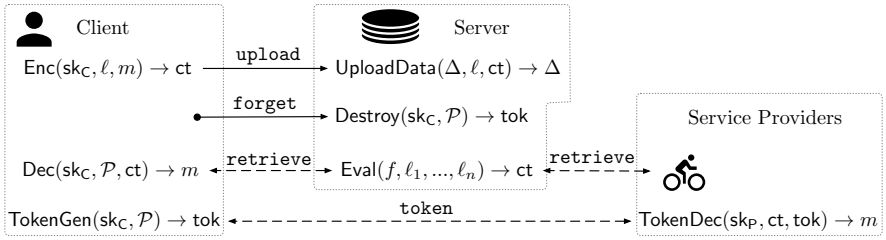


Figure 7: From Paper B: The HIKE protocol.

In particular, we focus on *verifiable random functions* (VRFs) and in particular on **simulatable VRFs** (sVRFs). In a nutshell, sVRFs are a family of VRFs in a public parameter security model, such as the common reference string.

Question C: Lattice sVRF: Challenges and Future Directions

> *Is it possible to define a* ***post-quantum*** *sVRF, based on lattice assumptions?*

Contribution: Paper C proposes the possibility of defining a **lattice-based membership hard with efficient sampling** language which can be used to define a lattice-based *dual-mode commitment scheme*. We partially conjecture the possibility to combine the dual-mode commitment scheme with Libert *et al.*'s protocol [LLNW17] and Lindell's transformation [Lin15] and obtain an sVRF under post-quantum assumptions, as represented in Fig. 8. Given the non-triviality of the task, we raise and identify different open challenges in lattice-based cryptography and possible future directions for achieving a post-quantum sVRF.

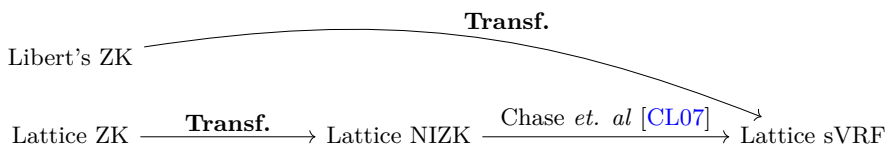


Figure 8: Paper C: A roadmap to lattice-based sVRF.

On a similar note, we ask ourselves:

Question D: Code-Based Zero Knowledge PRF Arguments

> *Is it possible to utilize a similar methodology as for Question C to define a* ***code-based*** *post-quantum zero-knowledge argument protocol?*

Contribution: Paper D utilizes the idea underlying Paper C by transforming a code-based PRG into a PRF for then introducing a methodology to effectively provide a zero knowledge argument for the code-based PRF evaluation. We propose a concrete construction and theoretically estimate the communication cost of our construction. Additionally, we introduce the *whistle-blower notary problem*, represented in Fig. 9, of which Paper C and D's results are possible solutions.

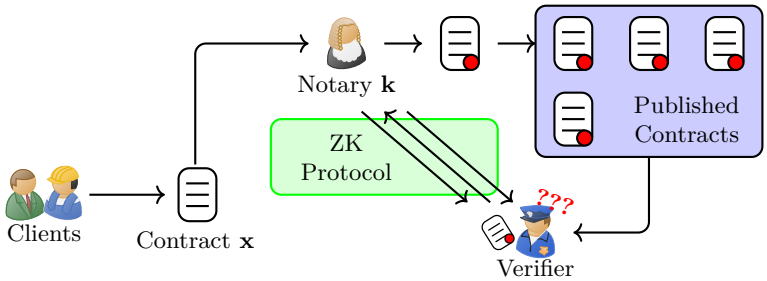


Figure 9: Paper D: The whistle-blower notary problem.

2.4 Verifying Functional Signature Evaluation

Signature schemes are a fundamental tool in today's application. They allow using a signing secret key to compute a signature from any message which later can be publicly verified with a public verification key and prove the authenticity of the content and the signer identity. A generalization of signature schemes is proposed by *Functional Signatures* (FS) in which the signer owns a *functional* signing key that allows signing a *specific function evaluation*. In other words, a functional signature allows authenticating the output of the function evaluation, therefore, hiding the original input.

An additional property provided by FS is *function hiding* in which it is impossible to infer which function got evaluated during the signature phase. In this way, verifying the signature correctness has two meanings: *(a)* the signature somehow verifies the correct evaluation of **a** function; and *(b)* the signature does not reveal **which** function got evaluated.

In a real application, often the signing key must be revoked which introduces a fundamental problem for FS: the function hiding property makes it impossible to know *which* signing key was used which means that the verification algorithm cannot effectively alert that a specific signature is generated from a revoked key.

Question E: Towards Stronger Functional Signatures

Is it possible to design a functional signature-like scheme that allows a more refined function evaluation verification ***but*** *preserves function privacy?*

Contribution: Paper E introduces the concept of *Strong Functional Signatures* (SFS), an FS-like scheme that introduces a public functional verification key that is publicly available and used during the verification phase. In a realistic application, such as the one represented in Fig. 10, all such public keys can be collected and publicly maintained by a trusted curator and allow key revocation by simply removing (or similar) the specific public key. SFS provides function hiding by requiring that both the signature **and** any functional verification public key hides which function is evaluated during the signing phase.

Our instantiation merges the Boneh-Lynn-Shacham's signature (BLS) scheme [BLS04] and Fiore-Gennaro's publicly verifiable computation (VC) scheme [FG12] under a shared *master* key pair used for the functional key generation and the final verification. Whenever generating the functional key pair, our instantiation first generates the VC's keys for the requested function and obtains the secret, evaluation and verification keys. Afterwards, the BLS's signing keys are generated with the addition of including additional information regarding the function **and** the VC's secret key. In this way, all the generated

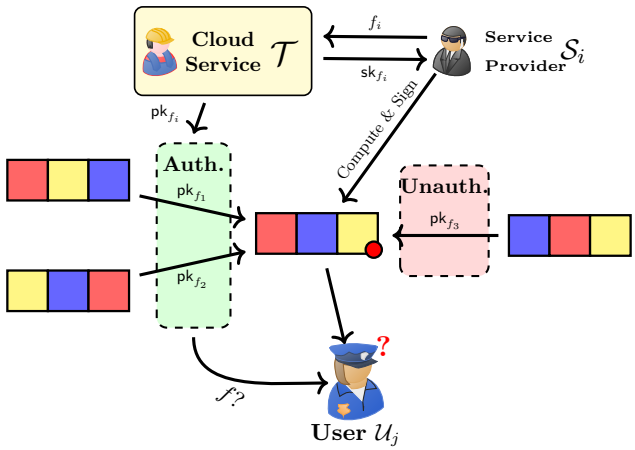


Figure 10: Paper E: Strong functional signatures in the cloud computational authentication scenario.

function's VC and BLS keys are related to each other.

The SFS's signing algorithm computes the VC evaluation and computes the BLS signature of the result which is later verified during the final verification. Our instantiation provides unforgeability by exploiting a design trick: a tamper must be a *"wrong evaluation"* which is signed with a BLS's key. Since the keys are all related, signing the wrong result will always create a wrong signature and if the BLS signature has correctly tampered with, then the tampered result must be the correct function evaluation which is not a tamper.

2.5 Machine Learning as a Tool for Cryptanalysis

Security is a complicated matter that can often be abstracted into *"hiding data's patterns"* while preserving some *"recovery"* property. Cryptanalysis is the research branch that applies several statistical, algorithmic and/or mathematical methodologies to find patterns in data to weaken or even destroy any security claim. The simplest form of such a methodology is based on solving a *distinguishing problem* in which an algorithm can classify the inputs between two (or more) different classes. The classical example is the *ciphersuite distinguishing problem* in which an algorithm takes in input a ciphertext and must output *"which is the encryption scheme used"*.

Machine Learning (ML) is a growing research area that provides a framework for investigating statistical correlations on specific datasets, often to extrapolate a classifier later used for analysing a new dataset.

Question F: Modelling Cryptographic Distinguishers Using Machine Learning

Can machine learning be used to automatize cryptanalysis?

Contribution: Paper F proposes an abstract methodology that allows to effectively use of ML for creating cryptographic distinguishers and provides some simple technique to improve the efficiency of such ML classifiers. Our methodology is depicted in Fig. 11.

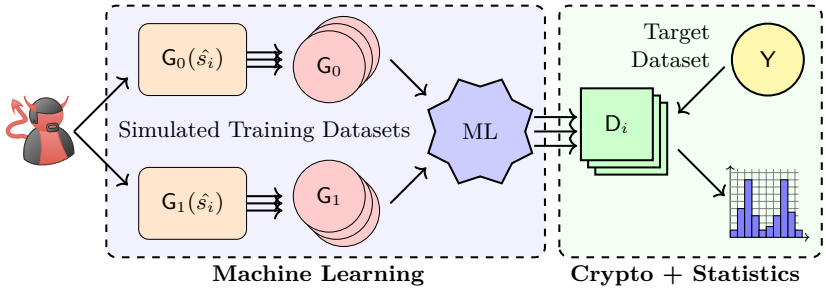


Figure 11: Paper F: Abstract representation of our methodology.

We implement our methodology in an expandable framework and create a simple proof-of-concept experiment in which we study the possibility of utilizing an ML generated distinguisher for distinguishing between several National Institute of Standard and Technology (NIST) Deterministic Random Bit Generators.

2.6 Secure Aggregation for Federated Learning

Federated Learning (FL) is a novel paradigm oriented to allow the aggregation of ML classifiers between several users with special consideration in achieving high privacy guarantees. The first privacy-preserving design concept is that each user pre-computes its ML model locally and it is not required to provide the raw data to the aggregating server. Only the computed model is used in the aggregation, therefore requiring the aggregation protocol to protect the user's model privacy.

Current solutions are focused on providing an interactive protocol between the users and a *single* central server that facilitates communication coordination. The interactivity of the protocol handles users that *drop out* from the protocol execution because either they lose their connection or they are maliciously trying to deny the service execution. Furthermore, the aggregating server is a *single-point-of-failure*. In an extreme scenario, an adversary might crash the central server and the protocol will abort without any recovery possibility.

Our specific interest is to additionally require the aggregating server to provide a proof that allows the users to verify the correctness of the servers computation.

Question G: Non-Interactive Secure Verifiable Aggregation for Decentralized, Privacy-Preserving Learning

> *Is it possible to distribute the secure aggregation between several servers* ***and*** *remove the necessity of the user's interaction* ***and*** *provide verification of the server evaluation correctness?*

Paper G proposes NIVA, a non-interactive primitive inspired by Shamir's secret sharing scheme that allows users to distribute the aggregation between several servers of which a threshold amount is needed to correctly reconstruct the final output, as depicted in Fig. 12 We implement NIVA and compare the communicational costs against some state-of-the-art protocol.

Contribution: our construction extends the standard additive homomorphic secret sharing scheme by introducing a *"verification token"* that the user computes and which is related to the secret input and the servers. During the aggregation phase, the servers compute and release the secret-sharing partial aggregation value and a proof of correct

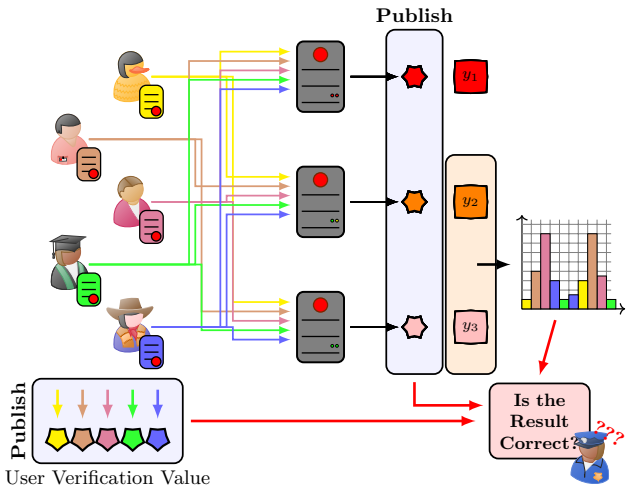


Figure 12: Paper G: Several users delegate the secure aggregation of their inputs to independent servers. A threshold amount of server's outputs is necessary to publicly reconstruct and verify the resulting aggregated value.

computation. The verification algorithm requires at least a threshold amount of server to be used to reconstruct the final aggregation **and** verify the computation correctness.

The confidentiality of the secret inputs is guaranteed by the underlying secret sharing scheme and the computational assumption used by the verification token. Differently, the scheme is proved to *never* be tamperable, *i.e.* any adversary is unable to provide a verifying wrong final aggregation result. The verification algorithm design allows to easily prove such a strong statement which boils down to an algebraic *"trick"*: the existence of an adversarial tamper depends on a pre-defined linear system which is easy to prove to **never** have a solution.

2.7 Alternative Communication Channels

The fundamental medium required for communicating is the *communication channel*. Different applications might require different *features*, *e.g.* we are interested in *consistent channels*. This means that the communication transcript is constantly verified during communication to prevent any *future* tampering of the *past* exchanged messages.

Blockchain is a novel technology that allows the creation of such a consistent channel. The only requirements are the *"complex"* assumptions necessary to create and use such a channel. Many blockchains require extensive use of signature schemes, public-key cryptography, hash functions and a *consensus mechanism*, often based on game-theoretic assumptions based on economical strategies.

Question H: Turn Based Communication Channel

Is it possible to create a consistent communication channel based on a minimal set of assumptions?

Paper H assumes the existence of a timed hash function, *i.e.* a hash function that is computable always in the same amount of time Δ. With such a primitive, we describe a *turn-based communication channel* (TBCC), depicted in Fig. 13

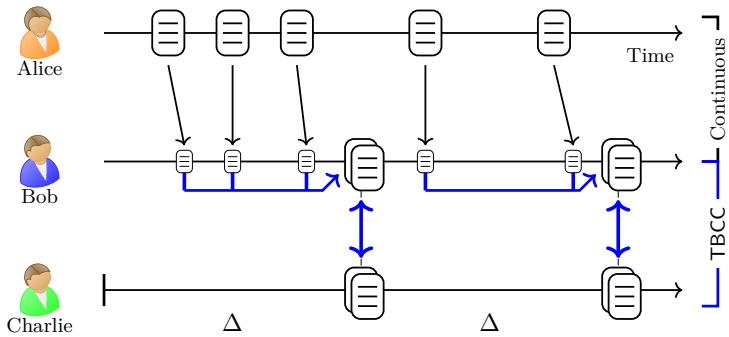


Figure 13: Paper H: A continuous and TBCC channel, the messages are gathered in "blocks", and each block, and its set of messages, is confirmed only at the end of each turn.

Contribution: we base our TBCC protocol on the idea of creating a verifiable *"commitment"* that can be verified only after solving a puzzle that requires a designed amount of time to be solved. Both the parties set up the communication by committing to a list of sequential puzzles which can only be solved in sequence. In this way, the parties start communicating committed messages that can only be periodically verified thus *emulating* a real turned communication where all the messages are exchanged periodically.

We provide a construction of the TBCC channel and prove that it provides communication consistency. This is possible because each exchanged message contains a *digest* of the previous communication thus making it impossible to tamper the communication without being noticed by the other party.

3 Summary and Future Directions

The papers contained in this thesis are testimony of the possibility of improving the crypto-toolset to incorporate privacy preservation and further allowing more secure solutions for real applications that requires to carefully handle people's sensitive data. Each one of the papers provides a novel cryptographic tool's instantiation that tackles a specific data leak, as summarised in Fig. 14.

Inevitably, data will increasingly be consumed by our evolving digital society and human understanding of data sensitivity will evolve accordingly, posing new security and privacy challenges to solve. For this reason, the research community **must** continue to develop new *verifiable* cryptographic tools that empower people and protect them from any harm caused by such a strong *data centricity*. Security and privacy are long-term requirements that must be incorporated in all the aspect of our society. More modest, shorter-term research directions would consider improvements such as:

- **Paper A** describes the possibility of easily introducing differential privacy in any cryptographic encryption scheme. On the other hand, it is left open the possibility to design different crypto-primitives that provides DP by design, *e.g.* would it be possible to create a DP signature scheme and which practical opportunities would it provide?

- **Paper B** provides a tailored GDPR-oriented solution called HIKE for a specific realistic application of outsourcing of both storage and computation. A direct

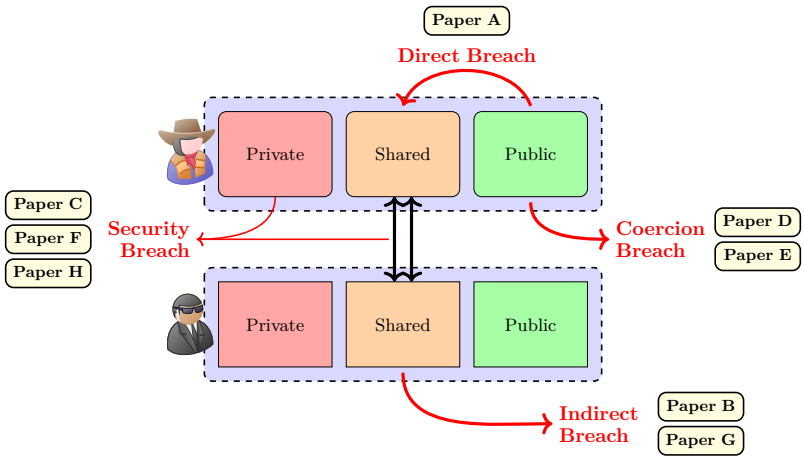


Figure 14: Paper's contributions and the correspondent data leak considered.

improvement would consist in either *(i)* further increasing HIKE's privacy requirements to cover more GDPR principles; *(ii)* simplifying the construction to improve efficiency; or *(iii)* introduce precise computation's verification requirements to guarantee security and privacy against stronger adversaries.

- **Paper C** and **Paper D** focus on the same goal of instantiating a simulatable verifiable random function. For both the papers, more research is necessary to allow them to be efficiently usable in practice. Additionally, different post-quantum cryptographic assumptions, *e.g.* isogenies, might be considered with the purpose of increasing the number of choices and allow the application to select the best-fitting primitive.

- **Paper E** provides the concept of strong functional signatures (SFS) and introduces an instantiation of SFS. A possible future direction would be to simplify the current instantiation, provide an efficient implementation and further investigate the possibility to define a general transformation that allows the instantiation of SFS from well-known cryptographic primitives.

- **Paper F** describes a methodology that enables the creation of crypto-distinguishers by utilising machine learning. It further provides an experimental analysis and an implementation. This paper's next step would be to improve the implementation by supporting additional machine learning algorithms and design a more practical and automatised framework. Of different motivation, it is of major interest the possibility to apply our methodology and check the concrete security of a real cryptographic system.

- **Paper G** introduces NIVA which is designed for federated learning's applications. Future directions would be focused on improving the primitive's efficiency and lowering the application requirements for secure usage of NIVA. From the practical side, NIVA should be implemented to be usable by the popular machine learning framework used by developers, *e.g.* TensorFlow.

- **Paper H** instantiates the concept of turn based communication channel (TBCC) and proves that the TBCC protocol achieves communication consistency. The

next step for TBCC would be to understand if it formally provides any cryptographic fairness property. Furthermore, a major investigation should be conducted to incorporate into the protocol more realistic assumptions, *e.g.* unpredictable communication delays.

A Differentially Private Encryption Scheme

Carlo Brunetta, Christos Dimitrakakis,
Bei Liang, and Aikaterini Mitrokotsa

Chalmers University of Technology, Gothenburg, Sweden

20-th Information Security Conference (ISC) 2017
Ho Chi Minh city (Viet Nam)

Abstract: Encrypting data with a semantically secure cryptosystem guarantees that nothing is learned about the plaintext from the ciphertext. However, querying a database about individuals or requesting for summary statistics can leak information. *Differential privacy* (DP) offers a formal framework to bound the amount of information that an adversary can discover from a database with private data, when statistical findings of the stored data are communicated to an untrusted party. Although both encryption schemes and differential private mechanisms can provide important privacy guarantees, when employed in isolation they do not guarantee full privacy-preservation.

This paper investigates how to efficiently combine DP and an encryption scheme to prevent leakage of information. More precisely, we introduce and instantiate *differentially private encryption schemes* that provide both DP and confidentiality. Our contributions are five-fold, we: *(i)* define an encryption scheme that is **not** correct with some probability α_{m_1,m_2} *i.e.,* an α_{m_1,m_2}-correct encryption scheme and we prove that it satisfies the DP definition; *(ii)* prove that combining DP and encryption, is equivalent to using an α_{m_1,m_2}-correct encryption scheme and provide a construction to build one from the other; *(iii)* prove that an encryption scheme that belongs in the DP-then-Encrypt class is at least as computationally secure as the original base encryption scheme; *(iv)* provide an α_{m_1,m_2}-correct encryption scheme that achieves both requirements (*i.e.,* DP and confidentiality) and relies on Dijk *et al.*'s homomorphic encryption scheme (EUROCRYPT 2010); and *(v)* perform some statistical experiments on our encryption scheme in order to empirically check the correctness of the theoretical results.

Keywords: Differential privacy, Encryption, Homomorphic encryption

1 Introduction

The Internet has evolved into a powerful platform interconnecting billions of users and has changed the way we do business, communicate with our friends, and perform our financial transactions. In this new communication paradigm, we leave our digital fingerprints everywhere: medical records, financial records, web search histories, and social network data. There is no doubt that the privacy implications of this increased connectivity can lead to oppressive electronic data surveillance.

Let us consider a real-world scenario: a company sells electricity to different customers in large geographical areas. The company owns and distributes a smart-metering grid [ETLP13] in order to offer the lowest price possible for its customers. Alice, that wants to pay as less as possible for her electrical consumption, signs a contract with the company by providing her personal information and accepts to install in her home different *sensors* that will measure the electrical consumption during the day and transmit this data to the electricity company. The company collects data from all its customers in an entire geographical region and, by performing statistical analysis on the collected data, is able to optimize the electrical supply distribution. Alice worries that her data may be used in a malicious way and wants to get guarantees that her privacy will be respected. She is aware that by analysing the data of her power consumption, someone may deduce private information such as when she is at home and what habits she may have. She wants her personal information to be confidential (encrypted) when they are used by a third party but she accepts that the company may use her data for statistical analysis in order to optimise the supply distribution.

This particular problem might raise different privacy concerns that we categorize into two classes, as represented in Figure (15):

- An *individual privacy breach* can be described as the act of deducing private information for an individual from some public information.
  In this case, the electricity company can deduce Alice's habits just by observing her power consumption measurements.

- A *group privacy breach* can be defined as the act of deducing a single individual private information from public statistical information of groups of people.
  Let us suppose that the electricity company offers an open-source interface where everyone can query and obtain statistical information about the company's customers. The only limitation is that the statistics are not computed if the sample of customers is lower than five people.
  Eve wants to find out Alice's habits for malicious reasons. To achieve that she checks on every social network and finds out that Alice is a *student* and she lives in a *one-room apartment*. Eve starts querying the company's database by asking for the *"average daily power consumption of a student that lives in an one-room apartment"* and does not obtain any information because the sample is too small. Then, Eve asks for the *"average daily consumption of people that live in an one-room apartment"* and the *"average daily consumption of people that live in an one-room apartment that are* ***not*** *students"*. Thus, Eve can deduce some approximation of Alice's habits by computing the difference between the two values and obtain the *"average daily consumption of a student that lives in an one-room apartment"* in which Alice is contained.

In this paper, we do not deal with the problem of inferring some private information about an individual (such as habits) from other private data, such as consumption, from a trusted third party (*e.g.,* a company). However, we care about inferring private information from publicly available data published by a third party (*e.g.,* the billing

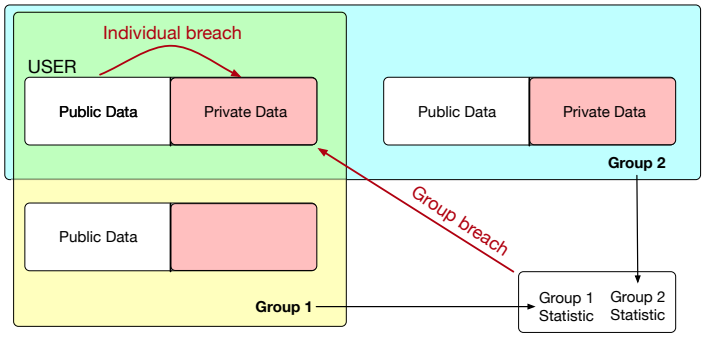


Figure 15: Individual and group privacy breaches.

information). To protect against either of the two types of privacy breaches, different notions of privacy and methodologies that preserve privacy have been defined in the literature such as *t-closeness* [LLV07], *k-anonymity* [EED08], *ℓ-diversity* [GKM11]. However, these notions of privacy have been proven to be weak, since even when they are employed information leakage and de-anomyization attacks can still be performed.

Differential privacy (DP) introduced by Dwork *et al.* [Dwo06], addresses the problem of learning as little as possible about an individual, while learning useful information about a population. It offers a formal framework that can be used to bound the amount of info that an adversary can discover from a database that contains private data, when statistical findings of the stored data are communicated to an untrusted party. More precisely, DP assumes the existence of a data aggregator, who is publishing statistics about a population. In other words, DP is a formalism that allows statistical analysis of private datasets while minimizing a group privacy breach. Informally, by employing a *DP-mechanism* to respond to a query, we are publishing noisy statistics about a dataset. The amount of noise should depend on the sensitivity of the queried statistic to the input, *i.e., "how much the query result would change if one single entry is changed or removed?"*. This means that if the query result will change a lot, we have to introduce more noise in order to *"hide"* the influence of the changed/removed entry in the query result. Otherwise, a drop in the query result will reveal partial information on the modified entry.

Complementary, a *semantically secure encryption* scheme guarantees the *confidentiality* of the encrypted information *i.e.,* no-one can decrypt and obtain the original message of a ciphertext. As a plus, an *homomorphic encryption* scheme [Gen09, Mei12] allows the computation of particular functions on the encrypted data. Informally, we can encrypt our messages and then compute a particular function on the ciphertext and obtain a new ciphertext that will be decrypted to the function computed on the original plaintext messages.

The solution required to avoid any possible information leakage should guarantee privacy breach *resistance* (provided by the DP framework) **and** confidentiality of the encrypted data (provided by a semantically secure encryption scheme). Each of these frameworks, if employed alone, does not provide full privacy guarantees. In this paper, we investigate for the first time, how we may achieve both differential privacy and confidentiality and introduce the concept of a differentially private encryption scheme.

Related Work: Privacy-preservation has received a lot of attention in the literature and multiple semantically secure crypto systems as well as differential private mechanisms have been proposed. However, existing work on encrypted computation and differential privacy has proceeded mainly in isolation. In order to avoid all possible

information leakage, while guaranteeing both *confidentiality* and *differential privacy*, the most common solution is to process the plaintext data in a DP-mechanism and then encrypt the result using a secure homomorphic encryption scheme. The ciphertext will guarantee confidentiality until the decryption phase, while the plaintext message will satisfy the DP definition. In the literature, it is possible to find different solutions [JLE14, BNO11, GJ11] that use this paradigm: a DP-mechanism and an encryption scheme; used sequentially. We will define these solutions that combine a *DP-framework* and an *Encryption-framework* as an element in the DP-then-Encrypt class (formally defined in Def. (5)). Our solution has as a starting point Dwork *et al.*'s definition of an α-correct encryption scheme [DNR04] *i.e.,* an encryption scheme that can *wrongly decrypt* (or encrypt) a message with some probability bounded by α. Dwork *et al.* [DNR04] defined an algorithm that takes an α-correct encryption scheme and returns a new encryption scheme, built using the α-correct one, that is correct (or almost-correct). We provide a more detailed definition of α-correctness, where we are interested in the precise probability of encrypting a message m_1 and obtaining a message m_2. Our definition is the first result that provides the sufficient conditions for an α-correct encryption scheme in order to achieve ϵ-DP. In order to build a concrete instantiation of a differentially private encryption scheme, we rely on Dijk *et al.*'s [vGHV10] homomorphic public-key encryption scheme over the integers.

Our Contributions: Our main idea is defining the class Encrypt+DP that contains all the encryption schemes that are differential private and achieve privacy and confidentiality *atomically*, as represented in Figure (16). As a starting point, we define an α_{m_1,m_2}-correct encryption scheme (Def. (4)) that will permit an encryption scheme to be **not** correct, *i.e.,* the decryption of the encryption of a specific message m_1 can be a different message m_2 with probability α_{m_1,m_2}. From this definition, we prove that an α_{m_1,m_2}-correct 1-bit encryption scheme satisfies the Dwork's DP definition [Dwo06] with $\epsilon(\alpha_{m_1,m_2})$-DP, *i.e.,* the DP parameter ϵ will be strongly related to the probabilities α_{m_1,m_2} of the encryption scheme. Then, we prove in Proposition (2) that the more general N-element encryption scheme achieves $\epsilon(\alpha_{m_1,m_2})$-DP.

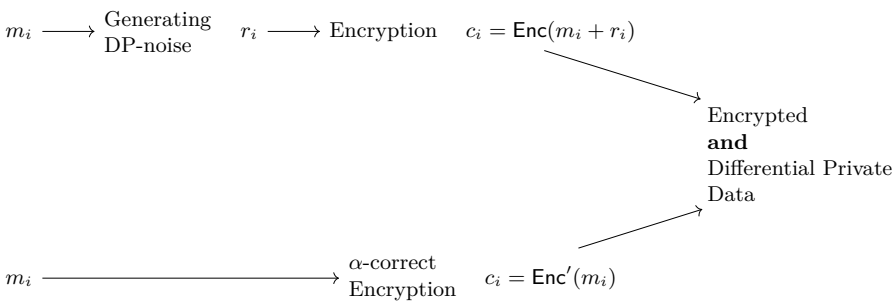


Figure 16: The difference between the DP-then-Encrypt (on the top) and our solution (at the bottom).

Furthermore, we formally define the DP-then-Encrypt and Encrypt+DP classes. As our main result, we prove in Proposition (4) that the two classes are equivalent and provide a construction to switch between them. This means that our solution of an α_{m_1,m_2}-correct encryption scheme can be re-written with a DP-then-Encrypt encryption scheme.

As the second main contribution, in Lemma 1, we reduce the security of a DP-

then-Encrypt encryption scheme to the security of the correct encryption scheme framework. The considered security-computational model is built around a non-interactive adversary that has access only to the public key and a particular ciphertext and it guesses the original plaintext. This security model is a necessary condition in order to satisfy more complex security models like IND − CPA, IND − CCA, etc.

The last contribution is a concrete $\alpha_{m,m}$-correct encryption scheme inside Encrypt+DP . We modify the Dijk *et al.* [vGHV10] integer homomorphic encryption scheme and we show how to compute the probability $\alpha_{m,m}$. As a final point, we exploit the structure of the scheme and obtain the correspondent DP-then-Encrypt encryption scheme that relies on Dijk *et al.*'s homomorphic encryption scheme.

Paper Organisation: The paper is organised as follows. In Section (2), we describe the notation used throughout the paper and the definitions we are based on. In Section (3), we give our definition of α_{m_1,m_2}-correct encryption schemes and prove that it has $\epsilon(\alpha_{m_1,m_2})$-DP. In Section (4), we show the equality between our framework, Encrypt+DP , and the DP-then-Encrypt . The proof will sketch an algorithm that transforms a correct encryption scheme into an α_{m_1,m_2}-correct encryption scheme. We define the security-hardness model and prove the security-hardness of a DP-then-Encrypt encryption scheme with respect to the corresponding base (correct) encryption scheme. In Section (5), we provide an instantiation of an $\alpha_{m,m}$-correct encryption scheme starting from Dijk *et al.*'s [vGHV10] encryption scheme and we prove its security.

2 Preliminaries

In this section, we will define the notation used in the paper and the basic definitions of the notions we employ in the rest of the paper.

2.1 Notation

We always denote with $\mathcal{M}$ the message-space. We denote with $\mathcal{K} = \mathcal{K}_{\mathsf{sk}} \times \mathcal{K}_{\mathsf{pk}}$ the key-space where $\mathcal{K}_{\mathsf{sk}}$ is the secret-key-space and $\mathcal{K}_{\mathsf{pk}}$ is the public key-space and with $\mathcal{C}$ the ciphertext-space. $\mathbb{N}$ is the set of natural numbers (*i.e.,* integers $z \geq 0$). Then we define intervals with $[a,b] = \{a, a+1, \cdots, b\}$ and $(a,b) = [a,b] \setminus \{a,b\}$. We denote with $\mathbb{1}_A$ the identity function on the set A. We define with the symbol $\simeq$, a *probabilistic* equality between functions, *i.e.,* $f(x) \simeq g(x)$ means $\mathsf{Pr}[f(x) = g(x)] = p$ for some $p \in [0,1]$. We denote with $\mathsf{negl}(n)$ a negligible function. We denote with $a \pmod n$ the modulo n of a in the interval $\left(-\frac{n}{2}, \frac{n}{2}\right]$. We denote with U_A the uniform distribution over the set A. We denote M times the cartesian product of a set A as A^M and the range of a function f with domain X as $\mathrm{Rg}(f) \coloneqq \{f(x) : x \in X\}$. For a set X, we define with $\mathcal{P}(X)$ the power-set of X, *i.e.,* the set of all the subset of X.

2.2 Basic Definitions

In order to define differential privacy, we will define a data-set:

Definition 1 (Dataset)**.** *A dataset D is defined on an alphabet A so that either $D \in A^n$ for a fixed dataset size n, or $D \in A^*$ with $A^* = \bigcup_{i=0}^{\infty} A^i$ being the union of all product sets of A.*

Definition 2 (ϵ-differential privacy [Dwo06])**.** *A randomized function $\mathcal{Q}$ is ϵ-differentially private if for all data-sets D_1 and D_2 differing on at most one element, i.e., the ℓ_0-distance between D_1 and D_2 is at most 1, and all $S \subseteq Rg(\mathcal{Q})$, it holds*

$$Pr[\mathcal{Q}(D_1) \in S] \leq \exp(\epsilon) \cdot Pr[\mathcal{Q}(D_2) \in S]$$

Remark 1. *For finite ϵ, we must have that the distribution of a DP-mechanism has always the same range,* i.e.*, for every $D_0, D_1 \subset \mathcal{M}$ it holds $Rg(\mathcal{Q}(D_0)) = Rg(\mathcal{Q}(D_1))$.*

In our construction, we will use messages as databases and we will always use the ℓ_0-distance; for two different messages m, m', the distance is always 1.

Below we provide Dwork *et al.*'s [DNR04] definition of an α-correct (public-key) encryption scheme:

Definition 3 (Dwork *et al.*'s α-correct public-key encryption scheme [DNR04])**.** *Let (G, E, D) be any public-key encryption scheme and $\alpha : \mathbb{N} \to [0, 1]$ an arbitrary function.*

(a) (G, E, D) *is* all-keys α-correct *if for every pair* $(\mathsf{sk}, \mathsf{pk})$ *generated by G on input 1^λ, it holds that $\mathsf{Pr}[D_{\mathsf{sk}}(E_{\mathsf{pk}}(m)) \neq m] \leq 1 - \alpha(\lambda)$, where the probability is taken over the choice of $m \in U_n$, and over the random coins of E and D.*

(b) (G, E, D) *is* almost-all-keys α-correct *if with probability $1 - \mathsf{negl}(\lambda)$ over the random coins of G used to generate* $(\mathsf{sk}, \mathsf{pk})$ *on input 1^λ, it holds that*

$$\mathsf{Pr}[D_{\mathsf{sk}}(E_{\mathsf{pk}}(m)) \neq m] \leq 1 - \alpha(\lambda)$$

where the probability is taken over the choice of $m \in U_n$ and over the random coins of E and D.

(c) (G, E, D) *is* almost-all-keys perfectly correct *if with probability $1 - \mathsf{negl}(\lambda)$ over the random coins of G used to generate* $(\mathsf{sk}, \mathsf{pk})$ *on input 1^λ, it holds that*

$$\mathsf{Pr}[D_{\mathsf{sk}}(E_{\mathsf{pk}}(m) \neq m] = 0$$

, where the probability is taken over the choice of $m \in U_n$ and over the random coins of E and D.

3 Our Definition of α_{m_1,m_2}-correct Encryption Scheme

In this section, we define an α_{m_1,m_2}-correct encryption scheme and compare it to the Dwork *et al.*'s Definition (3). Then, we prove that an α_{m_1,m_2}-correct encryption scheme satisfies the definition of differential privacy with respect to the function $\mathcal{Q} \coloneqq D \circ E \simeq \mathbb{1}_{\mathcal{M}}$. We start by presenting and describing the main constructions and properties for the case of a 1-bit encryption scheme, as the simplest example possible, and after that we generalize the result to an N-element encryption scheme.

3.1 Definition

Our goal is to formally define the possibility that an encryption scheme can wrongly decrypt a message with some well defined probability.

Definition 4 (α_{m_1,m_2}-correctness encryption scheme)**.** *Let (G, E, D) be an encryption scheme defined over $(\mathcal{M}, \mathcal{K}, \mathcal{C})$ as*

- ***Generation algorithm**: let $\lambda \in \mathbb{N}$ be a security parameter. G is defined as a probabilistic algorithm that given a security parameter 1^λ, returns a key-pair $(\mathsf{sk}, \mathsf{pk}) \in \mathcal{K}$.*
- ***Encryption algorithm**: let $m \in \mathcal{M}$, $\mathsf{pk} \in \mathcal{K}_{\mathsf{pk}}$ and $c \in \mathcal{C}$. E is defined as an algorithm that takes as input a public key pk and a message m, and returns a ciphertext c.*
- ***Decryption algorithm**: let $m \in \mathcal{M}$, $\mathsf{sk} \in \mathcal{K}_{\mathsf{sk}}$ and $c \in \mathcal{C}$. D is defined as an algorithm that given a secret key sk and a ciphertext c, returns a plaintext m.*

(G, E, D) is said to be an α_{m_1,m_2}-correct encryption scheme *if, for all $m_1, m_2 \in \mathcal{M}$, a fixed $\lambda \in \mathbb{N}$ and a fixed key-pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow G(1^\lambda)$, it holds*

$$\alpha_{m_1,m_2}((\mathsf{sk}, \mathsf{pk})) \coloneqq \mathsf{Pr}[D(\mathsf{sk}, E(\mathsf{pk}, m_1)) = m_2]$$

If for all $m \in \mathcal{M}$ it holds $\alpha_{m,m} = 1$, then (G, E, D) is said to be a correct *encryption scheme.*

In simple words, in an *α_{m_1,m_2}-correct encryption scheme*, the probability of encrypting m_1 and decrypting into m_2 using the key-pair $(\mathsf{sk}, \mathsf{pk})$ is equal to α_{m_1,m_2}.

Remark 2. *From the definition above, it is easy to see that every encryption scheme is an α_{m_1,m_2} encryption scheme.*

Remark 3. *The $\alpha_{m_1,m_2}((\mathsf{sk}, \mathsf{pk}))$ values are strongly connected with the choice of $(\mathsf{sk}, \mathsf{pk})$. We will abuse notation and drop the key-pair since in our arguments, we will always fix some key-pair $(\mathsf{sk}, \mathsf{pk})$.*

Remark 4. *Our α_{m_1,m_2}-correctness (Def. 4) and Dwork* et al.*'s definition (Def. 3) describe the same encryption schemes.*

Proof.

- *Our definition $\Rightarrow$ Dwork* et al.*'s definition*:
  Let (G, E, D) be any α_{m_1,m_2}-correct public-key encryption scheme. Let us consider
  $$\alpha = \max_{m \in \mathcal{M}, (\mathsf{sk},\mathsf{pk}) \in \mathcal{K}} \alpha_{m,m}((\mathsf{sk}, \mathsf{pk}))$$
  Let $(\mathsf{sk}, \mathsf{pk}) \in \mathcal{K}$ be any possible random key and $m \in \mathcal{M}$ any possible random message.
  $$1 - \mathsf{Pr}\left[D_{\mathsf{sk}}(E_{\mathsf{pk}}(m)) \neq m\right] = \mathsf{Pr}\left[D_{\mathsf{sk}}(E_{\mathsf{pk}}(m)) = m\right] = \alpha_{m,m}((\mathsf{sk}, \mathsf{pk})) \leq \alpha$$
  And so, we have that (G, E, D) is an α-correct encryption scheme in Dwork *et al.*'s Definition (3).
- *Our definition $\Leftarrow$ Dwork* et al.*'s definition*: Follows directly from Remark (2)

□

Dwork *et al.*'s definition describes a global upper bound on the correctness probability of an encryption scheme, while our definition defines the precise values of α_{m_1,m_2} of the encryption scheme.

3.2 Construction of an α_{m_1,m_2}-correct 1-bit Encryption Scheme

Fix $\mathcal{M} = \{0, 1\}$. Let (G, E, D) be an α_{m_1,m_2}-correct encryption scheme defined over $(\mathcal{M}, \mathcal{K}, \mathcal{C})$. Let us fix a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow G(1^\lambda)$ and let $\mathcal{Q}(m) = D(\mathsf{sk}, E(\mathsf{pk}, m))$. It holds:

$$\begin{gathered}
\mathrm{Rg}(\mathcal{Q}) = \{0, 1\} \qquad D_0 = \{0\}, D_1 = \{1\} \\
S \in \mathcal{P}(\mathrm{Rg}(\mathcal{Q})) = \{\emptyset, \{0\} = S_0, \{1\} = S_1, \{0, 1\} = \mathcal{M}\} \\
\mathcal{Q}(m) = D(\mathsf{sk}, E(\mathsf{pk}, m)) \simeq m \qquad \forall_{m_1,m_2 \in \mathcal{M}} \mathsf{Pr}\left[\mathcal{Q}(m_1) = m_2\right] = \alpha_{m_1,m_2}
\end{gathered}$$

Proposition 1. *An α_{m_1,m_2}-correct 1-bit encryption scheme such that for all $m_1, m_2 \in \mathcal{M}$ it holds that $\mathsf{Pr}[D(\mathsf{sk}, E(\mathsf{pk}, m_1)) = m_2] = \alpha_{m_1,m_2}$, achieves $\epsilon(\alpha_{m_1,m_2})$-differential privacy where*

$$\epsilon(\alpha_{m_1,m_2}) \coloneqq \inf \left\{ \epsilon : \begin{array}{c} e^\epsilon \geq \frac{\alpha_{0,0}}{\alpha_{1,0}} \;,\; e^\epsilon \geq \frac{\alpha_{0,1}}{\alpha_{1,1}} \\ e^\epsilon \geq \frac{\alpha_{1,0}}{\alpha_{0,0}} \;,\; e^\epsilon \geq \frac{\alpha_{1,1}}{\alpha_{0,1}} \end{array} \right\}$$

Proof. Let us prove that any α_{m_1,m_2}-correct encryption scheme satisfies the ϵ-DP definition.

From the Definition (2), we can state that $\mathsf{Pr}[\mathcal{Q}(D_i) \in S_j]$ means that we encrypt the bit i and we decrypt it into the bit j. We can impose the DP definition in all possible cases in order to study the differential privacy coefficient ϵ:

- If $S = \emptyset$, all the probabilities are 0, and so the ϵ-DP definition holds for every $\epsilon \in \mathbb{R}$ since $0 \leq 0$
- If $S = \{0,1\} = \mathcal{M}$, all the probabilities are 1, and so the ϵ-DP definition holds since $1 \leq e^\epsilon$ and $\epsilon \geq 0$
- If $S = \{0\} = S_0$:
  - $\mathsf{Pr}[\mathcal{Q}(D_0) \in S_0] \leq e^\epsilon \mathsf{Pr}[\mathcal{Q}(D_1) \in S_0]$ becomes $\alpha_{0,0} \leq e^\epsilon \alpha_{1,0} \Longrightarrow e^\epsilon \geq \frac{\alpha_{0,0}}{\alpha_{1,0}}$
  - $\mathsf{Pr}[\mathcal{Q}(D_1) \in S_0] \leq e^\epsilon \mathsf{Pr}[\mathcal{Q}(D_0) \in S_0]$ becomes $\alpha_{1,0} \leq e^\epsilon \alpha_{0,0} \Longrightarrow e^\epsilon \geq \frac{\alpha_{1,0}}{\alpha_{0,0}}$
- If $S = \{1\} = S_1$:
  - $\mathsf{Pr}[\mathcal{Q}(D_1) \in S_1] \leq e^\epsilon \mathsf{Pr}[\mathcal{Q}(D_0) \in S_1]$ becomes $\alpha_{1,1} \leq e^\epsilon \alpha_{0,1} \Longrightarrow e^\epsilon \geq \frac{\alpha_{1,1}}{\alpha_{0,1}}$
  - $\mathsf{Pr}[\mathcal{Q}(D_0) \in S_1] \leq e^\epsilon \mathsf{Pr}[\mathcal{Q}(D_1) \in S_1]$ becomes $\alpha_{0,1} \leq e^\epsilon \alpha_{1,1} \Longrightarrow e^\epsilon \geq \frac{\alpha_{0,1}}{\alpha_{1,1}}$

We can conclude that for every $\alpha_{m_1,m_2} \in [0,1]$, we achieve ϵ-DP where ϵ has to be in the convex solution set $\mathcal{E}(\alpha_{m_1,m_2})$ defined as:

$$\text{for } \alpha_{m_1,m_2} \in [0,1] \qquad \mathcal{E}(\alpha_{m_1,m_2}) \coloneqq \left\{ \epsilon : \begin{array}{ll} e^\epsilon \geq \frac{\alpha_{0,0}}{\alpha_{1,0}} & e^\epsilon \geq \frac{\alpha_{0,1}}{\alpha_{1,1}} \\ e^\epsilon \geq \frac{\alpha_{1,0}}{\alpha_{0,0}} & e^\epsilon \geq \frac{\alpha_{1,1}}{\alpha_{0,1}} \end{array} \right\}$$

from which we can define the curve

$$\epsilon(\alpha_{m_1,m_2}) = \inf \mathcal{E}(\alpha_{m_1,m_2})$$

that defines the minimum ϵ such that the ϵ-DP definition holds for the encryption scheme. □

Proposition (1) is a special case of Proposition (2).

3.3 Construction of an α_{m_1,m_2}-correct N-Elements Encryption Scheme

Let $\#\mathcal{M} = N$ be the message space with uniform distribution of being transmitted, *i.e.,* for all $m \in \mathcal{M}$, $\mathsf{Pr}[M \in \{m\}] = \frac{1}{\#\mathcal{M}}$. Fix a key-pair $(\mathsf{sk}, \mathsf{pk})$ and then for all $m_1, m_2 \in \mathcal{M}$ it holds

$$\alpha_{m_1,m_2} = \mathsf{Pr}[D(\mathsf{sk}, E(\mathsf{pk}, m_1)) = m_2 \mid m_1)$$

Proposition 2. *An N-element α_{m_1,m_2}-correct encryption scheme such that for all $m_1, m_2 \in \mathcal{M}$ it holds that $\mathsf{Pr}[D(\mathsf{sk}, E(\mathsf{pk}, m_1)) = m_2] = \alpha_{m_1,m_2}$. Then, the encryption scheme achieves $\epsilon(\alpha_{m_1,m_2})$-differential privacy where*

$$\epsilon(\alpha_{m_1,m_2}) \coloneqq \inf \left\{ \epsilon \;\middle|\; \forall D_0, D_1 \in \mathcal{M},\ S \subseteq \mathcal{M}.\ \frac{\sum_{m_2 \in S} \alpha_{D_0,m_2}}{\sum_{m_2 \in S} \alpha_{D_1,m_2}} \leq e^\epsilon \right\}$$

Proof. Let $\mathcal{Q} = D \circ E$ and $S \subseteq \mathcal{M}$ as before. Then, $\mathsf{Pr}[\mathcal{Q}(D_0) \in S] = \sum_{m_2 \in S} \alpha_{D_0,m_2}$.

Imposing the DP definition, we have that for all $D_0, D_1 \in \mathcal{M}$ such that the two elements are different and for every $S \subseteq \mathcal{M}$ it holds:

$$\mathsf{Pr}[\mathcal{Q}(D_0) \in S] \leq e^{\epsilon} \mathsf{Pr}[\mathcal{Q}(D_1) \in S] \implies \sum_{m_2 \in S} \alpha_{D_0,m_2} \leq e^{\epsilon} \left(\sum_{m_2 \in S} \alpha_{D_1,m_2} \right)$$

We can manipulate the equation and obtain $\frac{\sum_{m_2 \in S} \alpha_{D_0,m_2}}{\sum_{m_2 \in S} \alpha_{D_1,m_2}} \leq e^{\epsilon}$

We define the convex set

$$\mathcal{E}(\alpha_{m_1,m_2}) \coloneqq \left\{ \epsilon \;\middle|\; \forall D_0, D_1 \in \mathcal{M}, S \subseteq \mathcal{M}. \; \frac{\sum_{m_2 \in S} \alpha_{D_0,m_2}}{\sum_{m_2 \in S} \alpha_{D_1,m_2}} \leq e^{\epsilon} \right\}$$

The value $\epsilon(\alpha_{m_1,m_2}) = \inf \mathcal{E}(\alpha_{m_1,m_2})$ will satisfy the DP-definition. □

3.4 Fix ϵ, find α_{m_1,m_2}

The parameters ϵ and α_{m_1,m_2} are dependent one from the other since for all $D_0, D_1 \in \mathcal{M}$ and for all $S \subseteq \mathcal{M}$, it holds

$$\frac{\sum_{m_2 \in S} \alpha_{D_0,m_2}}{\sum_{m_2 \in S} \alpha_{D_1,m_2}} \leq e^{\epsilon} \tag{1}$$

The goal of finding the best α_{m_1,m_2} that achieves a fixed ϵ-DP depends on practical requirements and conditions that we want to impose on the probabilities α_{m_1,m_2}, *i.e., "maximizing the difference between two different messages"* or *"having a specific probability distribution"*.

For completeness, we will provide a simple solution in a particular case.

Proposition 3. *Let α_{m_1,m_2} be the probabilities of an N-element encryption scheme, where for all $m \in \mathcal{M}$, it holds $\alpha_{m,m} = \alpha$ and for all $m' \in \mathcal{M}$ with $m' \neq m$, it holds $\alpha_{m,m'} = \beta < \alpha$. If $\alpha \geq (N-1)\beta$, then the scheme achieves $\log\left(\frac{\alpha}{\beta}\right)$-DP.*

Proof. In order to prove the thesis, we have to find the D_0, D_1, S that maximize the left side of Equation (1). We can consider the polynomials $f_\alpha(x) = \alpha + x\beta$ and $f_\beta(x) = \beta + x\beta$. From the hypothesis, we have that $f_\alpha(x) \geq f_\beta(x)$ for all $x \in \mathbb{R}$ and $x \geq 0$. In particular, this is true for the integer values between 0 and $N-1$. Since $\frac{f_\alpha(x)}{f_\beta(x)}$ is a decreasing function for all $x \in \mathbb{R}$ and $x \geq 0$, we can conclude that for $i \in [0, N-1]$ integers, it holds:

$$\begin{aligned} \frac{\alpha}{\beta} = \frac{f_\alpha(0)}{f_\beta(0)} \geq \frac{f_\alpha(i)}{f_\beta(i)} \geq \frac{f_\alpha(i+1)}{f_\beta(i+1)} \geq \cdots \geq \frac{f_\alpha(N-1)}{f_\beta(N-1)} \\ \frac{\beta}{\alpha} = \frac{f_\beta(0)}{f_\alpha(0)} \leq \frac{f_\beta(i)}{f_\alpha(i)} \leq \frac{f_\beta(i+1)}{f_\alpha(i+1)} \leq \cdots \leq \frac{f_\beta(N-1)}{f_\alpha(N-1)} = \frac{(N-1)\beta}{(N-2)\beta + \alpha} \end{aligned} \tag{2}$$

From Equation (2) and since $\frac{\alpha}{\beta} \geq \frac{\beta}{\alpha}$ from the hypothesis, we have

$$\frac{(N-1)\beta}{(N-2)\beta + \alpha} \leq \frac{\alpha}{(N-2)\beta + \alpha} \leq \frac{\alpha}{\beta}$$

and, in Equation (1)

$$\frac{\sum_{m_2 \in S} \alpha_{D_0,m_2}}{\sum_{m_2 \in S} \alpha_{D_1,m_2}} \leq \frac{\alpha}{\beta} \leq e^{\epsilon} \tag{3}$$

We can so conclude that the minimal ϵ for which the equation holds is $\log\left(\frac{\alpha}{\beta}\right)$ and so the N-element encryption scheme will achieve $\log\left(\frac{\alpha}{\beta}\right)$-DP.

□

4 Equality Between DP-then-Encrypt and Encrypt+DP

In this section, we define the two main methods of combining an encryption scheme with a differential private mechanism: *(i)* the DP-then-Encrypt and *(ii)* the Encrypt+DP . We then prove a proposition on the equivalence between the DP-then-Encrypt and the Encrypt+DP classes. After this, we prove that combining a differential privacy framework with a correct encryption scheme is at least as computationally secure as the relying encryption scheme.

Definition 5. *Define the* DP-then-Encrypt *class as the set of all the encryption schemes* (G', E', D') *such that*

$$G'(1^\lambda) \coloneqq G(1^\lambda) \qquad E'(\mathsf{pk}, m) \coloneqq E(\mathsf{pk}, \mathcal{Q}(m)) \qquad D'(\mathsf{sk}, c) \coloneqq D(\mathsf{sk}, c)$$

for some (G, E, D) *correct encryption scheme on* $(\mathcal{M}, \mathcal{K}, \mathcal{C})$ *and* $\mathcal{Q} \simeq \mathbb{1}_{\mathcal{M}}$ *a DP-mechanism.*

It is trivial that $D'(\mathsf{sk}, E'(\mathsf{pk}, m)) = \mathcal{Q}(m)$.

Definition 6. *Define the* Encrypt+DP *class as the set of all the* α_{m_1,m_2}*-correct encryption schemes* $(\hat{G}, \hat{E}, \hat{D})$ *on* $(\mathcal{M}, \mathcal{K}, \mathcal{C})$*. From the Proposition* (2)*, we have that* $(\hat{G}, \hat{E}, \hat{D})$ *is* $\epsilon(\alpha_{m_1,m_2})$*-DP and it holds* $\hat{D}(\mathsf{sk}, \hat{E}(\mathsf{pk}, m)) \simeq \mathbb{1}_{\mathcal{M}}(m)$.

In a nutshell, the DP-then-Encrypt class contains all the different combinations of the identity map as a DP-mechanism and a correct encryption scheme. On the other hand, the Encrypt-then-DP achieves the identity map as a DP-mechanism directly in the α_{m_1,m_2}-correct encryption scheme used.

In order to prove the equality between the two classes, we define a *probability "permutation"* as:

Definition 7. *Let* $m_1, m_2 \in \mathcal{M}$*. Let us denote a* probability "permutation" π *as the random variable on* $\mathcal{M}$ *with measure probability of the event* "permute the message m_1 into the message m_2" *defined as* $\mathsf{Pr}[\pi(m_1) = m_2] = \alpha_{m_1,m_2}$.

Remark 5. *Let* π *be a probability permutation. Then,* π *is a DP-mechanism. This means it is a* $\epsilon(\alpha_{m_1,m_2})$*-DP mechanism (or it achieves* ∞*-DP).*

Proposition 4. *The* DP-then-Encrypt *class is equivalent to the* Encrypt+DP *class.*

Proof. • DP-then-Encrypt $\subseteq$ Encrypt+DP

Let (G', E', D') be a DP-then-Encrypt encryption scheme. Let us fix a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow G'(1^\lambda)$. Trivially using Remark (2), there exists an $\alpha_{m_1,m_2} \in [0, 1]$ such that for all $m_1, m_2 \in \mathcal{M}$ it holds:

$$\mathsf{Pr}\left[D'(\mathsf{sk}, E'(\mathsf{pk}, m_1) = m_2\right] = \mathsf{Pr}\left[\mathcal{Q}(m_1) = m_2\right] = \alpha_{m_1,m_2}$$

From the Definition (4), (G', E', D') is an α_{m_1,m_2}-correct encryption scheme and so from Proposition (2), we have that (G', E', D') is contained in the class Encrypt+DP of Definition (6).

- DP-then-Encrypt $\supseteq$ Encrypt+DP
  Let $(\hat{G}, \hat{E}, \hat{D})$ be an α_{m_1,m_2}-correct encryption scheme such that $\hat{D}(\mathsf{sk}, \hat{E}(\mathsf{pk}, m)) \simeq \mathbb{1}_{\mathcal{M}}(m)$. For every $m_1, m_2 \in \mathcal{M}$, we define the random variable $\pi : \mathcal{M} \to \mathcal{M}$ as

$$\mathsf{Pr}\left[\pi(m_1) = m_2\right] \coloneqq \mathsf{Pr}\left[\hat{D}(\mathsf{sk}, \hat{E}(\mathsf{pk}, m_1)) = m_2\right] = \alpha_{m_1,m_2}$$

  π is a probability permutation as in Definition (7) and for Remark (5), we have that π is a DP-mechanism.
  Let us define $(\hat{G}, E, D)$ a correct encryption scheme such that:

  - $\hat{G}$ is the same key generator as the α_{m_1,m_2}-correct encryption scheme
  - $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ is an encryption algorithm
  - $D : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$ is a decryption algorithm

  and for all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \hat{G}(1^\lambda)$, it holds that for all $m \in \mathcal{M}$

$$\mathsf{Pr}\left[D(\mathsf{sk}, E(\mathsf{pk}, m)) = m\right] = 1$$

  We can claim that E, D always exist and we can consider any injective function $\phi : \mathcal{M} \to \mathcal{C}$ with left inverse ϕ^{-1}. Let us define:

$$E(\mathsf{pk}, m) \coloneqq \phi(m) \qquad D(\mathsf{sk}, c) \coloneqq \phi^{-1}(c)$$

  For $(\hat{G}, E, D)$, we have

$$\mathsf{Pr}\left[D(\mathsf{sk}, E(\mathsf{pk}, m)) = m\right] = \mathsf{Pr}\left[\phi^{-1}(\phi(m)) = m\right] = \mathsf{Pr}\left[m = m\right] = 1$$

  In order to conclude, we need to prove that $(\hat{G}, E, D)$ with π as in Definition (5), acts like an encryption scheme (G', E', D') that is contained in the Encrypt+DP class of Definition (6). Fix a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \hat{G}(1^\lambda)$:

$$\begin{aligned}
\mathsf{Pr}\left[\hat{D}(\mathsf{sk}, \hat{E}(\mathsf{pk}, m_1)) = m_2\right] &= \alpha_{m_1,m_2} \\
&= \mathsf{Pr}\left[\pi(m_1) = m_2\right] \\
&= \mathsf{Pr}\left[\phi^{-1}(\phi(\pi(m_1))) = m_2\right] \\
&= \mathsf{Pr}\left[D(\mathsf{sk}, E(\mathsf{pk}, \pi(m_1))) = m_2\right] \\
&= \mathsf{Pr}\left[D'(\mathsf{sk}, E'(\mathsf{pk}, m_1)) = m_2\right]
\end{aligned}$$

□

We will now define a concept of *security-hardness* with respect to an adversary without specifying the computational model used.

Definition 8. *The adversary $\mathcal{A}$ for an encryption scheme (G, E, D) is an algorithm that takes the public key*[2] *and a ciphertext and it outputs a guess m' for the message m.*

$$\mathcal{A} : \mathcal{K}_{\mathsf{pk}} \times \mathcal{C} \to \mathcal{M} \qquad \mathcal{A}(\mathsf{pk}, E(\mathsf{pk}, m)) \mapsto m'$$

An encryption scheme (G, E, D) is said to be security-hard *with respect to the adversary $\mathcal{A}$ (in some computational model) if*

$$Pr[\mathcal{A}(\mathsf{pk}, E(\mathsf{pk}, m)) = m] \leq \frac{1}{\#\,\mathcal{M}} + \mathsf{negl}$$

[2]It is possible to give a pure symmetric key encryption scheme definition but we do not need it.

Informally, we defined the simplest adversary possible whose goal is to guess the correct decryption of a ciphertext given all the public information possible. In order to obtain a general result, we do not impose any complexity-hardness assumption. The security-hardness adversary is a weaker adversary with respect to the ones from IND−CPA, IND−CCA (and so on). On the other hand, for an encryption scheme, being security-hard is a necessary condition in order to achieve any security requirement: the security-hardness adversary can be used as a distinguisher in a more structured security model.

Lemma 1. *Let (G, E, D) be a correct encryption scheme which is security-hard. Let $\mathcal{Q} \simeq \mathbb{1}_{\mathcal{M}}$ DP-mechanism. Then the combination of $\mathcal{Q}$ with (G, E, D), which is in the* DP-then-Encrypt *class, is security-hard. In other word, the security-hardness of the combination $\mathcal{Q}$ with (G, E, D) is at least computationally hard as the security-hardness of (G, E, D).*

Proof. We have to show and prove:

(a) Reduce every instance of a (G, E, D) correct encryption scheme to an instance in the DP-then-Encrypt class.

(b) We prove the lemma by contradiction and *Reductio ad absurdum*: If there exists an adversary $\mathcal{A}$ with non-negligible advantage for the DP-then-Encrypt instance, there will exist an adversary $\mathcal{B}$ with non-negligible advantage for the (G, E, D) correct encryption scheme. Let us suppose that there exists $\mathcal{A}$ with non-negligible advantage, and let us suppose that all $\mathcal{B}$ have negligible advantage. Then we prove that it is a contradiction, and so we conclude.

The reduction is trivial: we can just consider as the instance in the DP-then-Encrypt class, (G, E, D) encryption scheme with the deterministic identity map as the DP-mechanism.

For a fixed key $(\mathsf{sk}, \mathsf{pk}) \leftarrow G(1^\lambda)$, suppose there exists an adversary $\mathcal{A}$ for the DP-then-Encrypt scheme, it means $\mathcal{A}(m) \coloneqq \mathcal{A}(\mathsf{pk}, E(\mathsf{pk}, \mathcal{Q}(m)))$ will output the guess m' and the guess will be correct with probability $\frac{1}{\#\mathcal{M}} + \delta$ with $\delta > 0$ non-negligible. Formally $\mathsf{Pr}\left[\mathcal{A}(\mathsf{pk}, E(\mathsf{pk}, \mathcal{Q}(m))) = m\right] = \frac{1}{\#\mathcal{M}} + \delta$

Let us suppose that for all the adversaries $\mathcal{B}$ of the original scheme such that $\mathcal{B}(m) \coloneqq \mathcal{B}(\mathsf{pk}, E(\mathsf{pk}, m))$, we have $\mathsf{Pr}\left[\mathcal{B}(\mathsf{pk}, E(\mathsf{pk}, m)) = m\right] = \frac{1}{\#\mathcal{M}} + \epsilon$ where $\epsilon > 0$ is negligible.

From the probability independence between the DP-mechanism $\mathcal{Q}$ and the encryption scheme (G, E, D) we have

$$\begin{aligned}\frac{1}{\#\mathcal{M}} + \delta = \mathsf{Pr}\left[\mathcal{A}(m) = m\right] &= \mathsf{Pr}\left[\mathcal{B}(m) = m \mid \mathcal{Q}(m) = m\right] \\ &= \mathsf{Pr}\left[\mathcal{B}(m) = m\right]\mathsf{Pr}\left[\mathcal{Q}(m) = m\right] \\ &\leq \mathsf{Pr}\left[\mathcal{B}(m) = m\right] = \frac{1}{\#\mathcal{M}} + \epsilon\end{aligned}$$

Absurd. So there exists an adversary $\mathcal{B}$ with non-negligible advantage.[3]

□

5 Example of an α_{m_1,m_2}-Correct Homomorphic Encryption Scheme

In this section, we introduce a variation of the Dijk's *et al.* public key integer homomorphic encryption scheme [vGHV10] by only introducing a new parameter ξ that will be used to increase the noisy randomness of the encryption scheme. Then, we show how

[3] Take for example adversary $\mathcal{A}$.

to compute the probabilities α_{m_1,m_2} that will prove that the scheme is α_{m_1,m_2}-correct. At the end, we show the connection between the original and the modified scheme and prove the security-hardness of the modified one.

Definition 9 (Variation of the Dijk *et al.* public key homomorphic encryption scheme)**.** *Let $\mathcal{M} = \{0,1\}$ and let γ, η, ρ, τ be the four parameters defined in the original scheme such that all the security constraints hold. Let ξ be an additional parameter required for the variation.*
Let (G, E, D) be defined as:

- $G(1^\lambda)$ *: randomly pick $p \in [2^{\eta-1}, 2^\eta)$ and p odd.*
  For the public key, for all $i \in 0..\tau$ sample

$$x_i \in \mathcal{D}_{\gamma,\rho}(p) = \left\{ pq + r : q \in U\left(\mathbb{Z} \cap \left[0, \frac{2^\gamma}{p}\right)\right), r \in U(\mathbb{Z} \cap (-2^\rho, 2^\rho)) \right\}$$

  and relabel so that x_0 is the greatest. Restart until x_0 is odd and $(x_0 \pmod p) \in \left(-\frac{p}{2}, \frac{p}{2}\right]$ is even.
  Define $\mathsf{pk} \coloneqq \{x_0, \dots, x_\tau\}$ as the public key and $\mathsf{sk} \coloneqq p$ as the secret key.

- $E(\mathsf{pk}, m)$*: choose at random $S \subseteq [1, \tau]$ and a random integer $r \in (-2^{\rho'+\xi}, 2^{\rho'+\xi})$. The difference with respect to the original scheme is that ξ is present in the interval-bounds exponents. Output the ciphertext $c = \left(m + 2r + 2\sum_{i\in S} x_i\right) \pmod{x_0}$*

- $D(p, c)$*: output $(c \pmod p) \pmod 2$*

In order to prove that the scheme achieves some α-correctness with $\alpha \neq 1$, fix a random S and observe that

$$\begin{aligned} m + 2r + 2\sum_{i\in S} x_i &= m + 2r + 2\sum_{i\in S} pq_i + r_i \\ &= m + 2\left(r + \sum_{i\in S} r_i\right) + p \cdot 2\sum_{i\in S} q_i = m + 2R + pQ \end{aligned}$$

where $Q \in \mathbb{Z}$ and R will be contained in a subset of the integers

$$A_S \coloneqq \left(-(\#S \cdot 2^\rho + 2^{\rho'+\xi}), (\#S \cdot 2^\rho + 2^{\rho'+\xi})\right) \subseteq \mathbb{Z}$$

For this reason, for a fixed S, we can reduce the computation of $\alpha_{m,m}$ as a combinatorial problem:

$$\alpha \coloneqq \frac{\#\left\{ r : r \in \left(-2^{\rho'+\xi}, 2^{\rho'+\xi}\right) \mid \left|2\left(r + \sum_{i\in S} r_i\right)\right| < \frac{p}{2} \right\}}{\#S \cdot 2^{\rho+1} + 2^{\rho'+\xi+1}}$$

For the right parameter ξ, we can obtain that the encryption scheme is an $\alpha_{m,m}$-correct encryption scheme.

Remark 6. *It is important to notice that using a different S will change the probability $\alpha_{m,m}$. You can think of it as* using a different public key *for the encryption algorithm.*

Consider a fixed S and the function $\lfloor x \rceil$ = closest integer to x. We can compute $\Delta = 2 \cdot \sum_{i\in S} r_i$ and if we consider $\tilde{\xi}$ as the bound for the noise r, we can define the function

$$F(\tilde{\xi}, \Delta) = \frac{\displaystyle\int_{-\tilde{\xi}+\Delta}^{\tilde{\xi}+\Delta} \left\lfloor \frac{x}{p} \right\rceil \pmod 2 \, dx}{2 \cdot \tilde{\xi}} \in [0, 1]$$

that represents the correctness probability. We have the trivial properties

$$F(\tilde{\xi}, 0) = \frac{1}{2} \qquad \lim_{\tilde{\xi}\to\infty} F(\tilde{\xi}, \Delta) = \frac{1}{2} \tag{4}$$

In order to prove that our modified scheme is secure, we reduce the security-hardness of our scheme to the security of the original Dijk *et al.*'s encryption scheme. From the Proposition (4) on the class equality between Encrypt+DP and DP-then-Encrypt we will transform our modified scheme into the Dijk *et al.*'s encryption scheme in the DP-then-Encrypt class.

Remark 7. *We can observe that r is randomly picked from $\left(-2^{\rho'+\xi}, 2^{\rho'+\xi}\right)$. We will now consider a random $r' \in (-2^{\rho'}, 2^{\rho'})$ and rewrite $r = r' + \hat{r}$ for some $\hat{r} \in \mathbb{Z}$. At this point, we can rewrite the general encrypted message as*

$$m + 2r + 2\sum_{i\in S} x_i = m + 2(r' + \hat{r}) + 2\sum_{i\in S} x_i = (m + 2\hat{r}) + 2r' + 2\sum_{i\in S} x_i \tag{5}$$

where r' and x_i are regular values from the original encryption scheme. During the decryption phase, we will obtain:

$$\begin{aligned}
&\left(m + 2r + 2\sum_{i\in S} x_i\right) \pmod p \pmod 2 = \\
&\qquad \textit{Equation (5)} = \left((m + 2\hat{r}) + \left(2r' + 2\sum_{i\in S} x_i\right)\right) \pmod p \pmod 2 \\
&\textit{Original scheme's values} = (m + 2\hat{r}) \pmod p \pmod 2 \\
&\qquad = m \oplus (2\hat{r} \pmod p \pmod 2)
\end{aligned}$$

From this equality, the message m can be decrypted in a different message $\hat{m}$ just by looking at the value $\hat{r}$.

This is exactly a DP-then-Encrypt scheme, where we can define a probability permutation π as in Definition (7) with $\mathsf{Pr}\left[\pi(m_1) = m_2\right] = \alpha_{m_1,m_2}$ and the original Dijk's encryption scheme.

Remark 8. *As in the Remark (6), changing S will change the probability permutation π since the probability α will change. For this reason, the random subset S, the probability permutation π, the probability α and the new parameter ξ are dependent one from the others.*

Proposition 5. *Given an $\alpha_{m,m}$-correct public key modified Dijk* et al. *'s encryption scheme with fixed parameters $(\rho, \rho', \eta, \gamma, \tau, \xi)$.*
Any adversary $\mathcal{A}$ with non-negligible advantage ϵ on the $\alpha_{m,m}$-correct encryption scheme can be converted into an adversary $\mathcal{B}$ with non-negligible advantage ϵ on the original Dijk et al.*'s encryption scheme with parameter $(\rho, \rho', \eta, \gamma, \tau)$.*

Proof. Follows from Lemma (1). □

5.1 Implementation and Statistics

In order to empirically study the dependency between the parameters ξ, α and ϵ, we implemented the modified Dijk *et al.* 's encryption scheme of Section 5 in Sage. Considering $\lambda = 10$ as a general security parameter, we started from the scheme with parameters:

$$\rho = \lambda \qquad \rho' = 2 \cdot \lambda \qquad \eta = \lambda^2 \qquad \gamma = \lambda^5 \qquad \tau = \lambda \qquad \xi = 0$$

and then we consider the k-th variation where we add a factor of $\tilde{\xi}_k = \frac{k \cdot p}{10}$ to the noise interval $2^{\rho'} + \tilde{\xi}_k$. In Figure 17, we have the measured value for α and ϵ with respect to k. For every $k \in [1, 30]$, we tested λ different choice of S, we executed $N = 100$ experiments and retrieved an empirical value for α. In order to obtain the ϵ, we just took the $\epsilon = \sup\left\{\frac{\alpha}{1-\alpha}, \frac{1-\alpha}{\alpha}\right\}$. We tested different random keys S and the empirical difference between the plots is barely visible, but it can easily be described as a *"really small translation of the plot to the left or right"*. In the chosen key used for the test, if we want to have a $\alpha = 0.8$ correctness probability, we have to use $\tilde{\xi}_4 = \frac{2 \cdot p}{5}$ and the scheme will have $\epsilon = 1.38$ -DP.

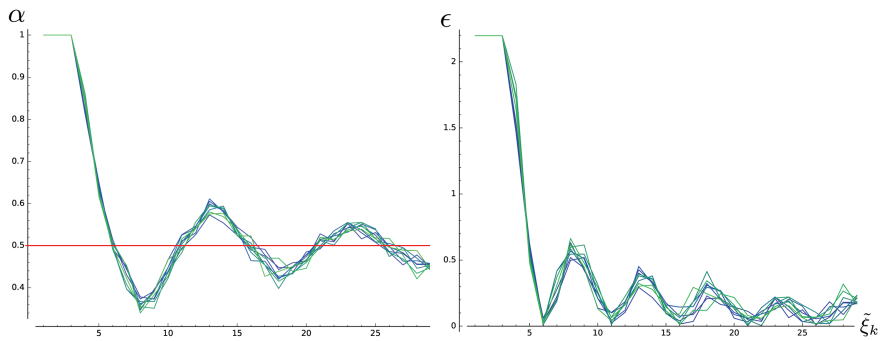


Figure 17: Empirical measurements of α and ϵ with respect to $\tilde{\xi}_k$.

6 Conclusions & Future Work

This paper bridges concepts in cryptography and differential privacy and we propose the first differentially private encryption scheme. More precisely, we show how to constructively combine differential privacy with an encryption framework in a single scheme, contained in the Encrypt+DP class, and vice versa. This construction is not limited to homomorphic encryption schemes and can be used in order to define an encryption scheme that can guarantee both privacy and confidentiality.

So far we have only examined this link in an abstract way. An open question is the trade-off between α_{m_1,m_2}-correctness and $\epsilon(\alpha_{m_1,m_2})$-DP for specific homomorphic operations, with a particular attention to the *bootstrap* procedure. This might lead to interesting practical applications, such as faster, α-correct homomorphic encryption schemes with differential privacy guarantees.

HIKE: Walking the Privacy Trail

Elena Pagnin, **Carlo Brunetta**, and Pablo Picazo-Sanchez

Chalmers University of Technology, Gothenburg, Sweden

17th International Conference on Cryptology And Network Security (CANS), 2018 Naples (Italy)

Abstract: We consider the problem of privacy-preserving processing of outsourced data in the context of user-customised services. Clients store their data on a server. In order to provide user-dependent services, service providers may ask the server to compute functions on the users' data. We propose a new solution to this problem that guarantees data privacy (*i.e.* an honest-but-curious server cannot access plaintexts), as well as that service providers can correctly decrypt only –functions on– the data the user gave them access to (*i.e.* service providers learn nothing more than the result of user-selected computations).

Our solution has as base point a new secure labelled homomorphic encryption scheme (LEEG). LEEG supports additional algorithms (FEET) that enhance the scheme's functionalities with extra privacy-oriented features. Equipped with LEEG and FEET, we define HIKE: a lightweight protocol for private and secure *storage*, *computation* and *disclosure* of users' data. Finally, we implement HIKE and benchmark its performances demonstrating its succinctness and efficiency.

Keywords: Homomorphic encryption, Privacy-preserving computation, Security protocol, GDPR

1 Introduction

We are living in the digital era, where people like to store their personal data in the cloud and get access to it any-time and anywhere. On the other hand, database maintainers and service providers develop an increasing interest for processing and extracting statistics from users's data. The usual setting is depicted in Figure 18: users (or clients) agree to share their personal data with some service providers which, in exchange, returns customised services and improved user-dependent performances. Typical application scenarios are: e-Health environments (*e.g.* keeping a blood pressure database that doctors can access to retrieve data) or smart trackers (*e.g.* activity bands that keep track of users' performance and achieved goals).

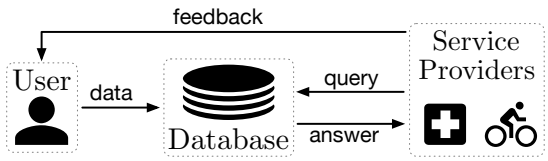


Figure 18: The setting we consider: users send data to a database and enjoy some service. *Example 1* (e-Health): doctors can query for the average blood pressure in the last hour and alert the user in case of need. *Example 2* (sport): the service provider can query for the distance run until 'now' and feedback when the daily goal is achieved.

In recent years, the cryptographic community has proposed new techniques for computing on outsourced data including Fully Homomorphic Encryption [Gen09], Verifiable Computation [GGP10] or Multi-Key Homomorphic Signatures [FMNP16]. Beyond the obvious benefits, user-customised services may have undesirable drawbacks. In particular, service providers can collect data from thousands of clients, identify trends, profile users, and potentially sell their knowledge to third parties without the clients' consent or awareness.

In this paper, we define a model for user-customised services that addresses new privacy challenges inspired to the guidelines provided in the European General Data Protection Regulation (GDPR) [Cou16]. This regulation sets clear boundaries on how data should be *collected, handled* and *processed* by protecting clients from possible miss-usages of their data by malicious service providers.

In particular, we give one of the *first attempts*[4] to *rigorously* formalise in *cryptographic* terms three of the main guidelines in the GDPR [Cou16], namely: (*i*) the client's data is never stored in plaintext on public databases (art. 32); (*ii*) the client decides who can read her data (art. 15); (*iii*) the client has the *right to be forgotten*, *i.e.* to request deletion of her data (art. 17).

Our Contributions. Our main contribution is the proposal and efficient instantiation of HIKE, a new cryptographic protocol that solves the problem of providing client-customised services. In details, our contributions are as follows:

(a) We present LEEG, a new labelled encryption scheme based on the elliptic-curve ElGamal scheme which supports homomorphic computation of multi-variate linear polynomials.

[4]The only academic works we found related to the GDPR are [Con18, PJ17], where the focus is on *technical* and *implementation* requirements. We could not find any work attempting to formalise and analyse the GDPR requirements in cryptographic terms.

(b) We define a set of additional algorithms that increase the versatility of LEEG, including an algorithm to cryptographically *destroy* encrypted data and a new procedure through which a chosen third party gets *decryption rights* for specific computations on encrypted data. We call this set of algorithms FEET as they extend the LEEG scheme.

(c) We then use LEEG and FEET in our HIKE protocol. HIKE is a novel lightweight protocol designed for application scenarios that involve users, servers, and service-providers. What makes this scenario different is that users' data need to be both privately and securely stored while allowing service providers to perform simple statistics on specific portions of users' data.

(d) We prove that HIKE is secure with respect to our security model that includes notions that address three articles of the GDPR law, namely (*i*) user's data is *never* stored as *plaintext* in the server; (*ii*) the user has the power to decide *who* can *read* its data; (*iii*) the user can always ask the server to cryptographically destroy its data.

(e) We implement the HIKE protocol and empirically test its succinctness and efficiency. We provide a complete benchmark for all the algorithms involved. Our implementation is freely available at https://github.com/Pica4x6/HIKE.

Overview of our Technique. Our starting point is the ElGamal encryption scheme on elliptic curves [Kob87, HPS14]. We progressively change this scheme by introducing three *ideas*: *(a)* replacing the sampling of randomness in a ciphertext by using labels and Pseudo Random Function (PRF); *(b)* modifying the labels to include the public key of the scheme, and; *(c)* exploiting the structure of the new ciphertexts to define algorithms for special user-privacy oriented features.

In more detail, but still quite abstractly, the three ideas work as follows. A label is a *unique identifier* for a specific message and it contains the sender's *public key*, a *random curve point* and a *tag* that identifies the message. Idea (i) is to change *how* the randomness is generated during the encryption procedure. We replace the random sampling of ElGamal encryption with the evaluation of a secure pseudo-random function $\mathsf{PRF}_{\mathsf{k}}$ on the label. For this change to work correctly, we also need to add the PRF key k to the user's secret key. The major implications of this change are: *1)* we can get rid of the random component of classical ElGamal ciphertexts (thus achieving better succinctness), and *2)* the new scheme has secret-key encryption.

Idea (ii) exploits the special structure of the labels and views the *"random curve point"* as the public key of the *designated-receiver* (*e.g.* the service provider). By doing so, we can algebraically manipulate ciphertexts in meaningful ways and also allow data decryption for both the encryptor and the *designated-receiver* (the latter upon receiving a special data-dependent *token*).

The last idea (iii) is to combine (i) and (ii) and design a protocol which addresses: data-secrecy (similar notion to semantic security); token-secrecy (data owner have full control on *who* can decrypt their data), and; forgettability (data owners can ask for their data to be destroyed).

Related Work. Rivest *et al.* [RAD78] introduced the concept of Homomorphic Encryption (HE) schemes as a set of algorithms that can be used to encrypt data, perform some computations on the ciphertexts, and directly decrypt the result of the computation.

For over 30 years, all secure proposals of HE schemes were only partially homomorphic, *i.e.* they supported either additions or multiplications of ciphertexts [Pai99,

ElG85]. The breakthrough result was due to Gentry [Gen09] and started an avalanche of Fully Homomorphic Encryption (FHE) schemes [SV10, BV14, SV10, CRRV17]. However, most FHE schemes have major drawbacks due to key sizes and (or) efficiency. Albeit HE supports less expressive computations than FHE, as long as we are interested in simple statistics (*e.g.* average, additions, least square fit of functions) on encrypted data, HE has better performances than FHE.

Barbosa *et al.* [BCF17] introduced the notion of Labelled Homomorphic Encryption (LabHE) which combines HE with labels. This is an elegant approach to address the problem of privacy-preserving processing of outsourced data. In this paper, we follow their definitional framework but we avoid the presence of a *fully trusted party* that executes the initial setup and holds a master secret key. Albeit being less expressive than Barbosa *et al.*'s scheme, our protocol achieves full succinctness without relaying on any trusted party.

A concurrent and independent work by Fischer *et al.* [FFKB17] proposes a linearly homomorphic construction also based on ElGamal encryption scheme. The aim of [FFKB17] is to provide both information flow security and authentication while our scheme has a privacy-oriented cryptographic approach —since we do not consider authentication, and it achieves full ciphertext succinctness.

2 Preliminaries

Notation. For any finite set S, we denote by $x \xleftarrow{\$} S$ the uniformly random sampling of elements from S, and by $|S|$ as the size of the set. We denote by $[n]$ the set $\{1, \dots, n\}$, by $[0..q]$ the set $\{0, \dots, q\}$, and by $\{0,1\}^*$ the space of binary-strings of arbitrary length. For any linear function f on n variables we describe f as $f(x_1, \dots, x_n) = a_0 + \sum_{i \in [n]} a_i x_i$, for opportune values $a_i \in [0..q-1]$. We denote by λ the security parameter of cryptographic schemes and functions, and by ε a negligible function in λ, *i.e.* $\varepsilon(\lambda) = O(\lambda^{-c})$ for every constant $c > 0$. We refer to computational feasibility (resp. infeasibility) of a problem if all known algorithm to solve the problem run in polynomial (resp. exponential) time.

Elliptic Curves. For prime p, let $\mathcal{E}$ be an elliptic curve over $\mathbb{F}_p$ and P be a generator point for the group $\mathbb{G}$ derived by $\mathcal{E}$. Let q be the order of $\mathbb{G}$, *i.e.* $\mathbb{G} =< P >= \{\mathcal{O}, P, 2 \cdot P, \dots, (q-1) \cdot P\}$, where $\mathcal{O}$ is the point at infinity (identity element of $\mathbb{G}$). For security reasons, we require q to be a prime number or a non-smooth (*i.e.* q is divisible by a large prime).

Problem 1 (Elliptic Curve Discrete Logarithm Problem [Kob87])**.** *Let p be a prime number and $\mathcal{E}$ be an elliptic curve over $\mathbb{F}_p$. Let $\mathbb{G}$ be the subgroup generated by a point $P \in \mathcal{E}$ such that $\mathbb{G} =< P >$ and $|\mathbb{G}| = q$ is prime or a non-smooth number. Given $Q \in \mathbb{G}$, the Discrete Logarithm (DLog) requires to find the value $m \in [0, \dots, q-1]$ such that $m \cdot P = Q$.*

Pollard's Rho [Pol78] is a well-known algorithm for solving the DLog problem. Its running time, however, is exponential in the group size, *i.e.*$O(\sqrt{|\mathbb{G}|}) = O(2^{\frac{q}{2}})$.

Assumption 1. *Given $\mathbb{G}$, P and Q as in Problem 1, it is computationally infeasible to find a solution to the DLog.*

Problem 2 (Interval Discrete Logarithm Problem [Pol00])**.** *Let $\mathcal{E}$, $\mathbb{F}_p$, $\mathbb{G}$, P and Q be as in Problem 1. The Interval Discrete Logarithm Problem (IDLP) requires to find the value $m \in [0, \dots, q-1]$ such that $m \cdot P = Q$ knowing that $m \in [a, \dots, b]$ for $a, b \in [0, \dots, q-1]$.*

Pollard's kangaroo algorithm [Pol00] finds an existing solution to the IDLP problem in a given interval $[a, \dots, b]$ in time $O(2^{\frac{\Delta}{2}})$ where $\Delta = \lceil \log_2(b-a) \rceil$ is the number of bits in the binary representation of the interval length [MT09].

Assumption 2. *Solving the IDLP is computationally feasible for* $|[a..b]| < 2^{22}$*, while it is infeasible for larger intervals* $|[a..b]| > 2^{160}$.

Pseudo Random Functions (PRF) [KL08]. A PRF is a collection of keyed functions from a (possibly infinite) set A to a finite set B. Formally, let $\mathcal{F}$ be the set of all functions from A to B and $\mathcal{K}$ be a (finite) set of keys, a PRF family is a set of functions $\{\mathsf{PRF}_\mathsf{k} : A \to B \mid \mathsf{k} \in \mathcal{K}\}$ satisfying the following properties:

(a) For any $a \in A$ and $\mathsf{k} \in \mathcal{K}$, the function $\mathsf{PRF}_\mathsf{k}(a)$ is efficiently computable.

(b) No Probabilistic Polynomial-Time (PPT) algorithm can distinguish the function PRF_k (for $\mathsf{k} \xleftarrow{\$} \mathcal{K}$) from a function $f \xleftarrow{\$} \mathcal{F}$.

In this paper, we regard HMAC-SHA256 as secure pseudo random function family for functions $\mathsf{PRF}_\mathsf{k} : \{0,1\}^* \to \mathbb{Z}_q$ with $\mathsf{k} \in \mathcal{K}$.

2.1 Labelled Homomorphic Encryption

The notion of labelled homomorphic encryption was introduced by Barbosa *et al.*to improve the efficiency of HE schemes [BCF17]. The main idea is to combine homomorphic encryption [Gen09] with labelled programs [GW13] to be able to compute on selected outsourced ciphertexts. A labelled program $\mathcal{P}$ is a tuple $(f, (\ell_1, \dots, \ell_n))$, such that $f : X^n \to X$ is a function of n variables and ℓ_i is a label for the i-th input of f. Labelled programs can be used to identify users' input to computations by imposing $\ell = (\mathsf{id}, \tau)$ for some user identifier id and tag τ [FMNP16]. We denote by $\mathcal{I}_\ell = (f, \ell)$ the *identity labelled program* on the label ℓ, *i.e.* $f_\ell(x) = x$.

Formally, a labelled homomorphic encryption scheme $\mathsf{LabHE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ is defined by the following algorithms:

$\mathsf{KGen}(1^\lambda)$: on input the security parameter, it outputs a secret key sk and a (public) evaluation key ek that includes a description of a message space $\mathcal{M}$, a label space $\mathcal{L}$, and a class of admissible functions $\mathcal{F}$.

$\mathsf{Enc}(\mathsf{sk}, \ell, \mathsf{m})$: on input sk, a label ℓ, and a message m, it outputs a ciphertext ct.

$\mathsf{Eval}(\mathsf{ek}, f, \mathsf{ct}_1, \dots, \mathsf{ct}_n)$: on input ek, a function $f : \mathcal{M}^n \to \mathcal{M}$ in a set of admissible functions $\mathcal{F}$, and n ciphertexts. It returns a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}, \mathcal{P}, \mathsf{ct})$: on input sk, a labelled program $\mathcal{P} = (f, \ell_1, \dots, \ell_n)$ and a ciphertext ct, it outputs a message m.

Moreover, LabHE satisfies the properties of correctness, succinctness, (semantic) security and context hiding defined in as follows.

Definition 10 (Correctness [BCF17])**.** *A* LabHE *scheme is said to be correct for a family of functions* $\mathcal{F}$ *if, for all keys* $(\mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$*, all* $f \in \mathcal{F}$*, any selection of labels* $\ell_1, \dots, \ell_n \in \mathcal{L}$*, and messages* $\mathsf{m}_1, \dots, \mathsf{m}_n \in \mathcal{M}$*, with corresponding ciphertexts* $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{sk}, \ell_i, \mathsf{m}_i)$*,* $i \in [n]$*, and* $\mathcal{P} = (f, (\ell_1, \dots, \ell_n))$*, it holds that:*

$$\mathsf{Prob}\left[\mathsf{Dec}\left(\mathsf{sk}, \mathcal{P}, \mathsf{Eval}(\mathsf{ek}, f, \mathsf{ct}_1, \dots, \mathsf{ct}_n)\right) = f(\mathsf{m}_1, \dots, \mathsf{m}_n)\right] \geq 1 - \varepsilon.$$

Definition 11 (Succinctness [BCF17])**.** *A* LabHE *scheme is said to be succinct if there exists a fixed polynomial* $\mathsf{poly}(\cdot)$ *such that every honestly generated ciphertext (output by* Enc *or* Eval*) has bit-size size* $\mathsf{poly}(\lambda)$.

The security notion for LabHE schemes is inspired to the standard semantic security experiment proposed by Goldwasser and Micali [GM82].

Definition 12 (Context Hiding [BCF17])**.** *A* LabHE *scheme is context-hiding if there exists a PPT algorithm* Sim *such that, for any* $(ek, sk) \leftarrow KGen(1^\lambda)$*,* $f \in \mathcal{F}$*, any tuple of labels* $\ell_1, \dots, \ell_n \in \mathcal{L}$ *and messages* $m_1, \dots m_n \in \mathcal{M}$ *with corresponding ciphertexts* $ct_1 \leftarrow Enc(sk, \ell_i, m_i)$*, if* $m = f(m_1, \dots, m_n)$ *and* $\mathcal{P} = (f, \ell_1, \dots, \ell_n)$ *then:*

$$\frac{1}{2} \cdot \sum_{ct} \left| Prob\left[Eval(ek, f, ct_1, \dots, ct_n) = ct\right] - Prob\left[Sim(1^\lambda, sk, \mathcal{P}, m) = ct\right] \right| < \varepsilon(\lambda)$$

Definition 13 (Semantic security for LabHE [BCF17])**.** *A* LabHE *scheme is semantically secure if for any PPT algorithm* $\mathcal{A}$ *taking part to the semantic security experiment* $Exp^{sem.sec}_{LabHE,\mathcal{A}}(\lambda)$ *in Figure 19, it holds that:*

$$Adv^{sem.sec}_{LabHE,\mathcal{A}}(\lambda) = Pr\left[Exp^{sem.sec}_{LabHE,\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} < \varepsilon.$$

$\underline{Exp^{sem.sec}_{LabHE,\mathcal{A}}(\lambda)}$:

$b \xleftarrow{\$} \{0,1\}, L_{lab} = \emptyset$
$(ek, sk) \leftarrow KGen(1^\lambda)$
$(\ell^*, m_0, m_1) \leftarrow \mathcal{A}^{OEnc_{sk}}(ek)$
if $\ell^* \in L_{lab}$
  $ct = error$
else
  $ct \leftarrow Enc(sk, \ell^*, m_b)$
  $L_{lab} \leftarrow L_{lab} \cup \ell^*$
$b' \leftarrow \mathcal{A}^{OEnc_{sk}}(ct)$
Output 1 if $b = b'$, and 0 otherwise.

$OEnc_{sk}(\ell, m)$:
if $\ell \in L_{lab}$
  return error
else $L_{lab} \leftarrow L_{lab} \cup \ell$
  return $ct \leftarrow Enc(sk, \ell, m)$.

Figure 19: The semantic-security experiment for LabHE schemes, and the OEnc oracle.

3 Labelled Elliptic-curve ElGamal (LEEG).

In its standard construction, the ElGamal encryption scheme [ElG85] is defined on finite multiplicative groups and is only multiplicative-homomorphic. It is possible to obtain an additive-homomorphic version of ElGamal by using groups defined over an elliptic curve [Kob87, HPS14] and specific message encoding maps, discussed later in Section 7. Essentially, in the elliptic curve setting, exponentiations are replaced by multiplications and multiplications by additions. Security reduces to the hardness of computing the discrete logarithm on elliptic curves. For further details on ElGamal for elliptic curve groups see [HPS14].

In this section, we define the first labelled and symmetric-key version of the additive-homomorphic ElGamal scheme that we refer to as LEEG (Labelled Elliptic-curve ElGamal). To ease the adoption of LEEG in our GDPR-oriented protocol HIKE in Section 5.1, we make a small adaptation to Barbosa *et al.*'s framework for LabHE. We introduce a Setup algorithm that outputs some global public parameters pp, and make the KGen algorithm run on pp. This change is only syntactic if KGen is run once and brings with straightforward modifications to the definitions.

Definition 14 (LEEG). *The* LEEG *scheme is defined by the following five PPT algorithms:*

Setup(1^λ)*: on input the security parameter , the setup algorithm outputs* pp *that include: a* λ*-bit-size prime* p*, an elliptic curve* $\mathcal{E}$ *over* $\mathbb{F}_p$ *with a (prime) order-*q *group* $\mathbb{G} \subseteq \mathcal{E}$*, a generator* P *of the group* $\mathbb{G}$*, a set of admissible functions* $\mathcal{F}$ *(namely linear functions), a set of messages* $\mathcal{M} \in [m]$*, a set of message identifiers* $\mathcal{T} = \{0,1\}^t$*, a set of labels* $\mathcal{L} = \mathbb{G} \times \mathbb{G} \times \mathcal{T}$*, and a keyed-pseudo-random function family* PRF *from* $\{0,1\}^*$ *to* $[0..q-1]$*. The* pp *are input to all subsequent algorithms even if not stated explicitly.*

KGen(pp)*: on input the public parameters the key generation algorithm selects a random element* $\mathsf{sk} \xleftarrow{\$} [q-1]$ *and a random PRF key* $\mathsf{k} \xleftarrow{\$} \mathcal{K}$*. It outputs the secret key* $\mathsf{sk} = (\mathsf{sk}, \mathsf{k})$.[5]

$\mathsf{Enc}(\mathsf{sk}, \ell, \mathsf{m})$*: on input a secret key* $\mathsf{sk} = (\mathsf{sk}, \mathsf{k})$*, a label* $\ell = (\mathsf{sk} \cdot P, Q, \tau)$ *with* $Q \in \mathbb{G}$ *and message* $\mathsf{m} \in \mathcal{M}$ *the encryption algorithm returns the ciphertext:*

$$\mathsf{ct} = \mathsf{m} \cdot P + \mathsf{PRF}_\mathsf{k}(\ell) \cdot \mathsf{sk} \cdot Q \in \mathbb{G} \tag{6}$$

In case the input label has as first entry a value different from $\mathsf{sk} \cdot P$ *the algorithm returns* error.

$\mathsf{Eval}(f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$*: on input a linear function* $f(x_1, \ldots, x_n) = a_0 + \sum_{i \in [n]} a_i x_i$ *and* n *ciphertexts* ct_i*, the evaluation algorithm returns the ciphertext:*

$$\mathsf{ct} = a_0 \cdot P + \sum_{i \in [n]} a_i \cdot \mathsf{ct}_i \in \mathbb{G} \tag{7}$$

$\mathsf{Dec}(\mathsf{sk}, \mathcal{P}, \mathsf{ct})$*: on input a secret key* $\mathsf{sk} = (\mathsf{sk}, \mathsf{k})$*, a labelled program* $\mathcal{P} = (f, \ell_1, \ldots, \ell_n)$ *for a linear function* f *with labels of the form* $\ell_i = (\mathsf{sk} \cdot P, Q_i, \tau_i)$*, and a ciphertext* ct*, the decryption algorithm computes:*

$$T = \mathsf{ct} - \mathsf{sk} \cdot \Big(\sum_{i \in [n]} a_i \cdot \mathsf{PRF}_\mathsf{k}(\ell_i) \cdot Q_i \Big) \in \mathbb{G} \tag{8}$$

and returns $\mathsf{m} = \log_P(T)$.

We note that LEEG is a fully dynamic scheme, indeed ciphertexts output by Eval can be used as input to new computations (as long as the new computation includes the initial labelled program).

Succinctness of LEEG. The succinctness of the LEEG scheme is immediate given that ciphertexts (output by Enc or Eval) are always one single group element in $\mathbb{G} \subseteq \mathcal{E}$. Further details regarding the actual bit-size for our implementation can be found in Section 7.

Correctness of LEEG. The correctness of LEEG is a straightforward computation.

Context-hiding of LEEG. The context-hiding property of LEEG is straightforward since given $\mathsf{sk}, \mathcal{P}$ and $\mathsf{m} = f(\mathsf{ct}_1, \ldots, \mathsf{ct}_n)$ the simulator is able to reconstruct exactly the same ciphertext output by $\mathsf{Eval}(f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$ as

$$\mathsf{Sim}(\mathsf{sk}, \mathcal{P}, \mathsf{m}) := f(\mathsf{m}_1, \ldots, \mathsf{m}_n) \cdot P + \mathsf{sk} \sum_{i \in [n]} a_i \mathsf{PRFk}(\ell_i) \cdot Q_i$$

[5] In the original definition of Labelled Homomorphic Encryption [BCF17], the KGen algorithm additionally outputs a public evaluation key. Since in our case this key is empty, we decided to skip it and have more succinct algorithm descriptions.

Security of LEEG. We want to prove that our LEEG scheme is semantically secure according to Barbosa *et al.*'s definition [BCF17] (Definition 13).

Proof. Let $Q_{\mathsf{prf}}(\lambda)$ be a bound on the total number of encryption queries performed by $\mathcal{A}$ during the security experiment. Let Game 0 be the semantic security experiment in Definition 13 where, for consistency with our definition of LEEG, the challenger runs $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{pp}$ to obtain the public parameters of the scheme, then it runs $\mathsf{KGen}(\mathsf{pp}) \rightarrow \mathsf{sk} = (\mathsf{sk}, \mathsf{k})$ and computes $\mathsf{pk} = \mathsf{sk} \cdot P$. In addition, $\mathcal{C}$ ignores any query with label $\ell = (Q', Q, \tau)$ where $Q' \neq \mathsf{pk}$.

Let Game 1 be the same as Game 0 except that the challenger replaces every $\mathsf{PRF}_{\mathsf{k}}(\cdot)$ instance with the evaluation of a truly random function $\mathsf{rand} : \mathbb{G} \times \mathbb{G} \times \mathcal{T} \rightarrow [0..q-1]$. It is quite easy to see that the difference between Game 1 and Game 0 is solely in the generation of the values r. Therefore the probability of $\mathcal{A}$ winning is the same in the two games, a part from a $Q_{\mathsf{prf}}(\lambda)$ factor that comes from distinguishing the PRF instance from a truly random function. Thus

$$|\mathsf{Prob}\left[\mathsf{G}_0(\mathcal{A})\right] - \mathsf{Prob}\left[\mathsf{G}_1(\mathcal{A})\right]| \leq Q_{\mathsf{prf}}(\lambda) \cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{A}}(\lambda).$$

At this point, we observe that for any given ciphertext ct and label-message pair (ℓ, m) there is exactly one value $r \in [0..q-1]$ for which ct is an encryption of m for label ℓ. In particular, for every triple $(\mathsf{ct}, \ell, \mathsf{m})$ it holds that

$$\mathsf{Prob}\left[\mathsf{Enc}(\mathsf{sk}, \ell, \mathsf{m}) = \mathsf{ct}\right] = \frac{|r \in [0..q-1] : M + r \cdot \mathsf{sk} \cdot Q = \mathsf{ct}|}{|[0..q-1]|} = \frac{1}{q},$$

where the probability is taken over all the possible values $r \xleftarrow{\$} [0..q-1]$. Since the above probability holds also for the challenge ciphertext ct^* we have that

$$\mathsf{Prob}\left[\mathsf{Enc}(\mathsf{sk}, \ell_0, \mathsf{m}_0) = \mathsf{ct}^*\right] = \mathsf{Prob}\left[\mathsf{Enc}(\mathsf{sk}, \ell_1, \mathsf{m}_1) = \mathsf{ct}^*\right]$$

(semantic security) and implies $\mathsf{Prob}\left[\mathsf{G}_1(\mathcal{A})\right] = \frac{1}{2}$. Therefore:

$$\begin{aligned}\mathsf{Prob}\left[\mathsf{G}_0(\mathcal{A})\right] &\leq |\mathsf{Prob}\left[\mathsf{G}_0(\mathcal{A})\right] - \mathsf{Prob}\left[\mathsf{G}_1(\mathcal{A})\right]| + \mathsf{Prob}\left[\mathsf{G}_1(\mathcal{A})\right] \\ &\leq Q_{\mathsf{prf}}(\lambda) \cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{A}}(\lambda) + \frac{1}{2}\end{aligned}$$

which proves the semantic security of LEEG, given that $Q_{\mathsf{prf}}(\lambda)$ is polynomial and $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{PRF}}(\lambda)$ is negligible (by our security assumption on the PRF family). □

4 FEET: Feature Extensions to the labelled homomorphic Elgamal encrypTion scheme

In this section we define FEET a set of additional algorithms that increase the versatility and use cases of LEEG. The new algorithms build on the following observation. Given a label $\ell = (Q_1, Q_2, \cdot)$ we can interpret its first component Q_1 as the public key of the user who is performing the encryption (that we call this data-owner), and Q_2 as the public key of another user (that we call intended receiver). By doing so, we can give a sensible meaning to the procedures and manipulate ciphertexts in such a way that decryption works correctly only for data encryptor statde in the first component of the label, and the *designated-receiver* identified by the second component of the label. Assuming the existence of a Public Key Infrastructure (PKI), FEET exploits the algebraic structure of LEEG ciphertexts to perform two actions:

- Cryptographically *'delete'* data owner's ciphertexts from a database by making them un-decryptable, *i.e.* even the original data-owner would retrieve a random message by decrypting a *destroyed* ciphertext.

- Allow the data-owner to generate a special piece of information (called *token*) that enables the intended receiver to decrypt the output of a specific labelled program run on the encryptor's data.

Definition 15 (FEET: set of additional algorithms for LEEG)**.** *Let* $\mathsf{LEEG} = (\mathit{Setup}, \mathit{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *be the labelled homomorphic encryption scheme of Definition 14, where the* KGen *algorithm is run multiple times and associates identities (identifiers* id*) to the keys it generates. We define:*

$\mathsf{Destroy}(\mathsf{ct})$**:** *on input a ciphertext* ct*, the destroy-cihpertext algorithm picks a random* $r \xleftarrow{\$} [0..q-1]$ *and outputs the destroyed ciphertext* $\mathsf{ct}' = \mathsf{ct} + r \cdot P$*.*

$\mathsf{PublicKey}(\mathsf{sk})$**:** *on input the secret key* $\mathsf{sk} = (\mathit{sk}, \mathsf{k})$ *output the corresponding public key* $\mathsf{pk} = \mathit{sk} \cdot P$*.*

$\mathsf{TokenGen}(\mathsf{sk}, \mathcal{P})$**:** *on input a secret key* $\mathsf{sk} = (\mathit{sk}, \mathsf{k})$ *and a labelled program* $\mathcal{P} = (f, \ell_1, \dots, \ell_n)$ *the token generation algorithm checks if the labels are of the form* $\ell_i = (\mathit{sk} \cdot P, Q, \tau_i)$ *— for a point* $Q \in \mathbb{G}$ *and some* $\tau_i \in \mathcal{T}$*,* $i \in [n]$*. If the condition is not satisfied, the algorithm returns* error*; otherwise, it parses* $f(x_1, \dots, x_n) = a_0 + \sum_{i \in [n]} a_i x_i$ *and outputs :*

$$\mathsf{tok} = \mathit{sk} \cdot \Big(\sum_{i \in [n]} a_i \cdot \mathsf{PRF}_{\mathsf{k}}(\ell_i) \Big) \cdot P \tag{9}$$

$\mathsf{TokenDec}(\mathsf{sk}, \mathsf{ct}, \mathsf{tok})$**:** *on input a secret key* $\mathsf{sk} = (\mathit{sk}, \mathsf{k})$*, a ciphertext* ct *and a token* tok*, the decryption-with-token algorithm outputs* $\mathit{m}' = \log_P (\mathsf{ct} - \mathit{sk} \cdot \mathsf{tok})$*.*

Information theoretic security of Destroy**.** We prove this property by showing that for any given message m and ciphertext ct', ct' is a possible 'destruction' of $\mathsf{ct} = \mathsf{Enc}(\mathsf{sk}, \ell, \mathsf{m})$ for any label. More formally, for any $\mathsf{m} \in \mathcal{M}$, $\ell = (\mathsf{sk} \cdot P, \mathsf{pk}, \tau)$ and $\mathsf{ct}' \in \mathbb{G}$ it holds that:

$$\mathsf{Prob}\left[\mathsf{Destroy}(\mathsf{Enc}(\mathsf{sk}, \ell, \mathsf{m})) = \mathsf{ct}'\right] = \frac{|\{r : \mathsf{ct}' = (\mathsf{m} + r') \cdot P + \mathsf{sk} \cdot r \cdot \mathsf{pk}\}|}{|\mathbb{G}|} = \frac{1}{|\mathbb{G}|}$$

where the probability is taken over all possible random choices in the $\mathsf{Destroy}$ algorithm ($r' \in [0..q-1]$), and $r = \mathsf{PRF}_{\mathsf{k}}(\ell)$. In particular, for any pair of label-message couples $(\ell_0, \mathsf{m}_0), (\ell_1, \mathsf{m}_1)$ it holds that:

$$\mathsf{Prob}\left[\mathsf{Destroy}(\mathsf{Enc}(\mathsf{sk}_{\mathsf{id}_0}, \ell_0, \mathsf{m}_0)) = \mathsf{ct}'\right] = \mathsf{Prob}\left[\mathsf{Destroy}(\mathsf{Enc}(\mathsf{sk}_{\mathsf{id}_1}, \ell_1, \mathsf{m}_1)) = \mathsf{ct}'\right].$$

Therefore given a ct' output by $\mathsf{Destroy}$ this could be generated by the ciphertext of *any* message $\mathsf{m} \in \mathcal{M}$.

Correctness of TokenDec**.** In order to prove the correctness of the decryption-with-token algorithm we need to show that

$$\mathsf{TokenDec}(\mathsf{sk}_2, \mathsf{Eval}(f, \mathsf{ct}_1, \dots, \mathsf{ct}_n), \mathsf{TokenGen}(\mathsf{sk}_1, \mathcal{P})) = f(\mathsf{m}_1, \dots, \mathsf{m}_n)$$

where $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{sk}_1, \ell_i, \mathsf{m}_i)$, $\ell_i = (\mathsf{pk}_1, \mathsf{pk}_2, \tau_i)$ for some $\tau_i \in \mathcal{T}$, and $\mathcal{P} = (f, \ell_1, \dots, \ell_n)$. This is a straightforward computation and follows from Equation (7), Equation (9) and the fact that $\mathsf{pk}_i = \mathsf{sk}_i \cdot P$.

Remark 9 (Composability of TokenGen)**.** *It is possible to combine two (or more) decryption tokens* $\mathsf{tok}_1, \mathsf{tok}_2$ *generated for distinct labelled programs* $\mathcal{P}_1, \mathcal{P}_2$*, to obtain a joint decryption token* tok *that enables the intended decryptor with public key* $\mathsf{pk}_2 = Q$ *(common to all the labels involved) to correctly decrypt in one-shot the ciphertext for any labelled program* $\mathcal{P}$ *such that* $f = b_1 f_1 + b_2 f_2$*, for any* $b_1, b_2 \in [0..q-1]$*.*

Consider two labelled programs $\mathcal{P}_1 = (f_1, \ell_1, \dots, \ell_n)$ *and* $\mathcal{P}_2 = (f_2, \ell'_1, \dots, \ell'_{n'})$. *For consistency, token composability requires that all the labels involved in* $\mathcal{P}_1$ *and* $\mathcal{P}_2$ *are of the form* $\ell = (\mathsf{sk} \cdot P, Q, \tau)$ *for some opportune value of* τ. *Without loss of generality, we can set* $f_1 = a_0 + \sum_{i\in[n]} a_i x_i$ *and* $f_2 = a'_0 + \sum_{j\in[n']} a'_j x_{\psi(j)}$ *for opportune coefficients* $a_i, a'_j \in \mathcal{M}$, *and an index-mapping function* $\psi : \mathbb{N} \to \mathbb{N}$ *used to model the fact that the functions may be defined on a different set of variables. Let* $I \subseteq \mathbb{N}$ *be the set of indexes of common variables, formally:*

$$I = \{i \in [n] \text{ such that } \psi(j) = i \text{ for some } j \in [n']\}. \tag{10}$$

The composed labelled program $\mathcal{P} = b_1\mathcal{P}_1 + b_2\mathcal{P}_2$ *is defined as* $\mathcal{P} = (f, \tilde{\ell}_1, \dots, \tilde{\ell}_{\tilde{n}})$ *with* $f = b_1 f_1 + b_2 f_2$, $f(x_1, \dots, x_{\tilde{n}}) = (b_1 a_0 + b_2 a'_0) + \sum_{i\in I}(b_1 a_i + b_2 a'_i)x_i + \sum_{i\in[n]\setminus I} b_1 a_i x_i + \sum_{j\in[n']\setminus\psi(I)} b_2 a'_j x_j$ *for any* $b_1, b_2 \in \mathcal{M}$. *We show that the combined token* $\mathsf{tok} = b_1\mathsf{tok}_1 + b_2\mathsf{tok}_2$ *is a valid decryption token for the composed labelled program* $\mathcal{P}$, *actually* $\mathsf{tok} = \mathsf{TokenGen}(\mathsf{sk}, \mathcal{P})$. *In details:*

$$\begin{aligned}
\mathsf{tok} &= b_1\mathsf{tok}_1 + b_2\mathsf{tok}_2 \\
&= \mathit{sk}\Big(\sum_{i\in[n]} b_1 a_i r_i + \sum_{j\in[n']} b_2 a'_j r'_{\psi(j)}\Big) \cdot P \\
&= \mathit{sk}\Big(\sum_{i\in I}(b_1 a_i + b_2 a'_i) r_i + \sum_{i\in[n]\setminus I} b_1 a_i r_i + \sum_{j\in[n']\setminus\psi(I)} b_2 a'_j r'_j\Big) \cdot P \\
&= \mathsf{TokenGen}(\mathsf{sk}, \mathcal{P})
\end{aligned}$$

The set I *in the second last equality is the one defined in* (10), *that is* $\ell_i = \ell'_{i=\psi(j)}$ *for all* $i \in I$ *and therefore* $r_i = r'_i = \mathsf{PRF}_{\mathsf{k}}(\ell_i)$. *By the correctness of the* TokenGen-TokenDec *algorithms, we derive that* tok *is a valid decryption token for* sk_2, $\mathsf{ct} = \mathsf{Eval}(f, \mathsf{ct}_1, \dots, \mathsf{ct}_{\tilde{n}})$. *It is straightforward to generalise this reasoning to multiple labelled programs* $\mathcal{P}_1, \dots \mathcal{P}_t$ *as long as all the labels coincide on the first two entries.*

5 The HIKE protocol

In this section, we introduce HIKE: a protocol that employs our LEEG scheme and its extra features FEET to achieve advanced properties relevant to real world applications.

In what follows, we present a use case for HIKE and defer the formal description to the Section 5.1. We consider a scenario with three types of actors: data-owners (called clients and denoted as $\mathcal{U}$), a cloud server (denoted as $\mathcal{S}$) that controls the database Δ where the clients' records are stored, and service providers (denoted as $\mathcal{T}$). The workflow of the interactions between these actors is depicted in Figure 20. As a use case, consider clients with smart-watches used for tracking their sport performances. With HIKE clients can safely upload their data on the cloud, cancel previously uploaded records, retrieve their data (or functions of thereof) at any time. Moreover, clients can allow their personal trainer app to access specific aggregations of data to provide personalised performance feedback.

5.1 A formal description

Definition 16 (The HIKE protocol)**.** *Let* LEEG = (*Setup*, *KGen*, Enc, Dec, Eval) *be the labelled homomorphic encryption scheme in Definition 14 enhanced with the algorithms* FEET = (TokenGen, TokenDec, Destroy) *described in Definition 15. We assume a PKI that, at every run of the* *KGen* *algorithm, associates identities (identifiers* id*) to the freshly generated keys. The* HIKE *protocols is defined by the following procedures:*

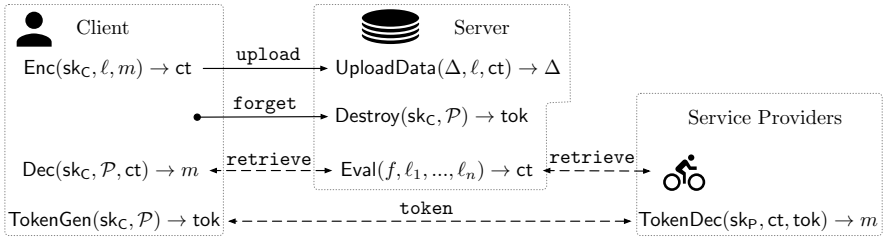


Figure 20: Clients upload their encrypted data to the server (via an upload request). At any point in time, clients have the right to *destroy* their records in the database (via a forget request). In order to obtain aggregate information on the stored data, clients and service providers can ask the server to compute certain functions on the outsourced data and return the (encrypted) result (via a retrieve request). Finally, clients are always able to decrypt their own retrieved data, while service providers cannot decrypt directly. In order to decrypt the result of a computation $\mathcal{P} = (f, \ell_1, \dots, \ell_n)$ on the client's data, the service provider needs to ask the client to generate a computation-specific decryption token (via a token request) that enables the designated service provider to decrypt.

$\mathsf{Initialise}(1^\lambda)$*: on input the security parameter* λ*, the initialisation procedure runs* $\mathit{Setup}(1^\lambda) \to \mathsf{pp}$ *and returns the public parameters. Implicitly, it also generates a database* Δ *and a public key infrastructure.*

$\mathsf{SignUp}(\mathsf{id})$*: on input a user identifier* id *the sign-up procedure returns* $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathit{KGen}(\mathsf{pp})$ *and updates the public ledger (PKI) with* $(\mathsf{id}, \mathsf{pk}_{\mathsf{id}})$ *where* $\mathsf{pk}_{\mathsf{id}} = \mathsf{PublicKey}(\mathsf{sk}_{\mathsf{id}})$*. For correctness, this procedure outputs* $\perp$ *if user* id *was already present in the system.*

$\mathsf{Encrypt}(\mathsf{sk}, \ell, m)$*: on input a secret key* sk*, a label* ℓ *and a message* m *the encryption procedure returns the ciphertext* $\mathsf{ct} = \mathsf{Enc}(\mathsf{sk}, \ell, m)$*.*

$\mathsf{UploadData}(\Delta, \ell, \mathsf{ct})$*: on input a database* Δ*, a label* ℓ *and a ciphertext* ct *the upload data procedure performs* $\Delta = \Delta \cup \{(\ell, \mathsf{ct})\}$*.*

$\mathsf{Forget}(\Delta, \ell)$*: on input a database* Δ *and a label* ℓ *the forget-ciphertext procedure retrieves the record* (ℓ, ct) *from* Δ *and replaces it with* (ℓ, ct') *where* $\mathsf{ct}' \leftarrow \mathsf{Destroy}(\mathsf{ct})$*,* i.e.*it outputs* $\Delta = \Delta \setminus \{(\ell, \mathsf{ct})\} \cup \{(\ell, \mathsf{ct}')\}$*.*

$\mathsf{Compute}(\Delta, \mathcal{P})$*: on input a database* Δ *and a labelled program* $\mathcal{P} = (f, \ell_1, \dots, \ell_n)$ *the retrieve-data (or aggregate data) procedure collects the ciphertexts* ct_i *corresponding to labels* ℓ_i *present in* Δ *and returns* $\mathsf{ct} = \mathsf{Eval}(\mathsf{pp}, f, \mathsf{ct}_1, \dots, \mathsf{ct}_n)$*.*

$\mathsf{Decrypt}(\mathsf{sk}, \mathcal{P}, \mathsf{ct})$*: on input a secret key* sk*, a labelled program* $\mathcal{P}$*, and a ciphertext* ct *the decryption procedure outputs* $m = \mathsf{Dec}(\mathsf{sk}, \mathcal{P}, \mathsf{ct})$*.*

$\mathsf{AllowAccess}(\mathsf{sk}, \mathcal{P})$*: on input a user's secret key* sk *and a labeled program, the allow-access procedure returns* $\mathsf{tok} = \mathsf{TokenGen}(\mathsf{sk}, \mathcal{P})$*.*

$\mathsf{AccessDec}(\mathsf{sk}, \mathsf{ct}, \mathsf{tok})$ *on input a user's secret key* sk*, a ciphertext* ct *and a decryption token* tok*, the allowed-decryption procedure returns the output of* $\mathsf{TokenDec}(\mathsf{sk}, \mathsf{ct}, \mathsf{tok}) = m$*.*

5.2 Evaluation correctness of HIKE

The evaluation correctness of our HIKE protocol essentially reduces to the correctness of the underlying LEEG scheme (Definition 14) and FEET (Definition 15). Formally, the HIKE is correct if for any $\mathsf{pp} \leftarrow \mathsf{Initialise}(1^\lambda)$, for any combination of keys $(\mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U})$, $(\mathsf{pk}_\mathcal{T}, \mathsf{sk}_\mathcal{T})$ generated by the SignUp procedure, for any labelled program $\mathcal{P} = (f, \ell_1, \dots, \ell_n)$ with labels for the form $\ell = (\mathsf{sk}_\mathcal{U} \cdot P, Q = \mathsf{pk}_\mathcal{T}, \cdot)$, for any set of messages $\mathsf{m}_i \in \mathcal{M}$ with ciphertexts $\mathsf{ct}_i = \mathsf{Encrypt}(\mathsf{sk}, \ell_i, \mathsf{m}_i)$, and for $\mathsf{m} = f(\mathsf{m}_1, \dots, \mathsf{m}_n)$ it holds that:

(1) $\mathsf{Decrypt}(\mathsf{sk}_\mathcal{U}, \mathcal{P}, \mathsf{Compute}(\mathcal{P})) = \mathsf{m}$.

(2) $\mathsf{AccessDec}\left(\mathsf{sk}_\mathcal{T}, \mathsf{Compute}(\mathcal{P}), \mathsf{AllowAccess}(\mathsf{sk}_\mathcal{U}, \mathcal{P})\right) = \mathsf{m}$.

Condition *(1)* is equivalent to the evaluation correctness of the LEEG scheme given that the Compute procedure returns the output of LEEG's Eval algorithm and the Decrypt procedure runs LEEG's Dec algorithm.

Condition *(2)* is equivalent to the correctness of LEEG's additional algorithms given that AllowAccess returns the output of TokenGen, Compute returns the output of LEEG's Eval algorithm and AccessDec runs the TokenDec algorithm.

5.3 Interactions between the procedures of HIKE

We consider three categories of users taking part to the HIKE protocol:

Clients $\mathcal{U}$: (or data owners), these users can run the procedures: SignUp, Encrypt, Decrypt and AllowAccess.

Server $\mathcal{S}$: (or database maintainer), this user can run the procedures: UploadData, Compute and Forget.

Service providers $\mathcal{T}$: (or third-party applications), these users can run the procedures: SignUp and AccessDec.

To simulate a real-world scenario, we allow users registered in the system to interact with each other. We model interaction via requests sent from one user to another and that there exists an authentication system to ensure this. Moreover, we assume that the target user reacts to the received request as follows:

upload: upload data requests can be performed by clients only and are directed to the server. Upon receiving an $\mathsf{upload}(\ell, \mathsf{ct})$ request by a client $\mathcal{U}$, the server checks if the submitted record is a new one , *i.e.* if $(\ell, \cdot) \notin \Delta$. In this is the case, $\mathcal{S}$ runs $\mathsf{UploadData}(\Delta, \ell, \mathsf{ct})$ and returns done to $\mathcal{U}$, otherwise $\mathcal{S}$ returns error.

forget: forget-ciphertext requests can be performed by clients only and are directed to the server. Upon receiving an $\mathsf{forget}(\ell)$ request by a client $\mathcal{U}$, the server checks that the label is a legit one for $\mathcal{U}$, *i.e.* that $\ell = (\mathsf{pk}_\mathcal{U}, \cdot, \cdot)$ and that $(\ell, \cdot) \in \Delta$. If both conditions holds, $\mathcal{S}$ runs $\mathsf{Forget}(\Delta, \ell)$ and returns done to $\mathcal{U}$, otherwise it returns error.

retrieve: retrieve-data requests can be performed by clients or by service providers and are directed to the server. Upon receiving a $\mathsf{retrieve}(\mathcal{P})$ request the server checks if the labelled program $\mathcal{P} = (f, \ell_1, \dots, \ell_n)$ is well-defined, *i.e.* if for every $i \in [n]$, $(\ell_i, \cdot) \in \Delta$ and $\ell_i = (\mathsf{pk}_h, \mathsf{pk}_k, \cdot)$ for some users registered in the systems. If the conditions hold, $\mathcal{S}$ runs $\mathsf{Compute}(\Delta, \mathcal{P}) = \mathsf{ct}$ and returns ct to whom performed the query, otherwise it returns error.

token: access-token requests can be performed by service providers only and are directed to clients only. Upon receiving a $\mathsf{token}(\mathcal{P})$ request, the client has the freedom to decide whether to reply consistently, *i.e.* running the algorithm $\mathsf{AllowAccess}(\mathsf{sk}, \mathcal{P}) = \mathsf{tok}$ and returning tok to $\mathcal{T}$, or to ignore the query returning error.

6 Security model and proofs for HIKE

Our security model builds on the setting introduced in Section 5.3 and covers three main goals:

1) data-secrecy, *i.e.* confidentiality of the clients' data;

2) token-secrecy, *i.e.* clients have full control on *who* can decrypt their data (only targeted service providers can decrypt, and no one else); and

3) forgettability, *i.e.* clients can ask for their data to be destroyed.

Interestingly, these security notions cover three of the requirements presented in the GDPR: the confidentiality of personal data (security of processing, art. 32), the clients' right of access (and share) data (art. 15), and; the right to ask for erasure of her personal data (*right to be forgotten*, art. 17) [Cou16].

Adversarial model We denote malicious users with the user's category and the symbol *, *e.g.* $\mathcal{T}^*$, and make the following assumptions:

- Clients $\mathcal{U}_i$ are *honest*, *i.e.* they behave according to the interactions described in Section 5.3.
- The server $\mathcal{S}$ is *honest but curious*, *i.e.* it behaves according to the interactions in Section 5.3 but tries to infer information about the clients' data (passive adversary).[6]
- Service Providers $\mathcal{T}_j$ can be *malicious* and deviate from the protocol in arbitrary ways.

We note that since anyone can generate and register keys in the protocol (using the PKI infrastructure), a malicious server corresponds to a malicious service provider that has access to an honest server (as the latter would reply to any $\mathsf{retrieve}$ request).

6.1 Data secrecy

Our notion of data-secrecy is inspired to the definition of semantic-security for labelled homomorphic encryption by Barbosa *et al.* [BCF17] but adapted to our protocol's setting. Intuitively, we require that the adversary $\mathcal{A}$, who controls a (malicious) server provider $\mathcal{T}^*$ and holds a copy of the (encrypted) database Δ, should not be able to determine the plaintext associated to a database record (ℓ, ct).
We formalise the notion through the data-secrecy experiment in Figure 21.

Notably, $\mathcal{A}$ is given access to three oracles: $O\mathsf{SignUp}$ to simulate users registering to the system, $O\mathsf{Encrypt}$ to populate the database with adversarial chosen data (*i.e.* $\mathcal{A}$ chooses the messages encrypted by a client). With abuse of notation we will write $\mathsf{pk}_{\mathsf{id}} \in L_{\mathsf{keys}}$ meaning that L_{keys} has an element of the form $(\mathsf{id}, \mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}})$.

[6] This assumption removes the theoretical need for the definition of forgettability in our security model. We include it for completeness.

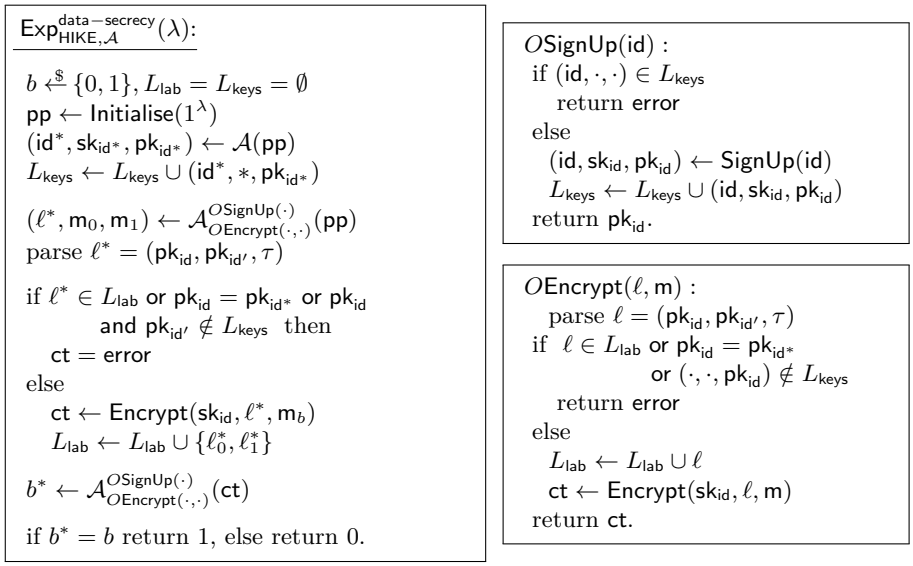


Figure 21: The data-secrecy experiment and the oracles $O\mathsf{SignUp}$ and $O\mathsf{Encrypt}$.

Theorem 1. *The* HIKE *protocol achieves data-secrecy,* i.e. *for any PPT adversary* $\mathcal{A}$ *taking part to the experiment in Figure 21, it holds that:*

$$\mathsf{Adv}^{\mathsf{data.sec}}_{\mathsf{HIKE},\mathcal{A}}(\lambda) = Pr\left[Exp^{\mathsf{data.sec}}_{\mathsf{HIKE},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \leq Q_{\mathsf{id}} \cdot \mathsf{Adv}^{\mathsf{sem.sec}}_{\mathsf{LEEG},\mathcal{A}}(\lambda),$$

where Q_{id} *is a bound on the total number calls to the sign-up oracle.*

Proof. We exhibit a reduction $\mathcal{B}$ that uses $\mathcal{A}$ to win the semantic-security experiment for the LEEG scheme. At the beginning the reduction samples $\mathsf{id}^\star \in \mathsf{ID}$ as its guess for the identity that $\mathcal{A}$ will target during the game. (Note that $|\mathsf{ID}| = Q_{\mathsf{id}}$ is polynomial in this game). This step corresponds to $\mathcal{B}$ betting that $\mathcal{A}$ will choose the client $\mathsf{id}^\star$ in its challenge labels.

When the semantic-security experiment starts, the reduction $\mathcal{B}$ (that is playing as an adversary) gets $\mathsf{ek} = (\mathsf{pk}^\star = \mathsf{sk} \cdot P, \mathsf{pp})$ from its challenger $\mathcal{C}$. Then $\mathcal{B}$ starts the data-secrecy experiment (as a challenger) by sending pp to $\mathcal{A}$. The adversary chooses an identity id^* and a pair of keys for it. $\mathcal{A}$ also sends $(\mathsf{id}^*, \mathsf{pk}_{\mathsf{id}^*})$ to $\mathcal{B}$. The reduction registers the (malicious) user id^* in the system ($L_{\mathsf{keys}} \leftarrow L_{\mathsf{keys}} \cup (\mathsf{id}^*, \cdot, \mathsf{pk}_{\mathsf{id}^*})$) and replies to $\mathcal{A}$'s queries as follows.

sign-up queries: $\mathcal{B}$ forwards the queries to the $O\mathsf{SignUp}$.

encryption queries: $\mathcal{B}$ forwards the queries to the $O\mathsf{Encrypt}$ oracle unless $\ell = (\mathsf{pk}^\star, \mathsf{pk}, \tau)$, in which case $\mathcal{B}$ updates the list of queried labels $L_{\mathsf{lab}} \leftarrow L_{\mathsf{lab}} \cup \ell$, forwards (ℓ, m) as an encryption query to $\mathcal{C}$ and relays its reply to $\mathcal{A}$.

Let $(\ell^*, \mathsf{m}_0, \mathsf{m}_1)$ be $\mathcal{A}$'s input to the challenge phase. If $\ell^* \neq (\mathsf{pk}^\star, \cdot, \cdot)$ the reduction aborts (as $\mathcal{A}$ chose to challenge a different client than the one $\mathcal{C}$ has created). This event happens with probability $(1 - \frac{1}{Q_{\mathsf{id}}})$. Otherwise $\ell^* = (\mathsf{pk}^\star, Q, \tau)$, $\mathcal{B}$ updates the list of queried labels $L_{\mathsf{lab}} \leftarrow L_{\mathsf{lab}} \cup \ell^*$ and sends $(\ell^*, \mathsf{m}_0, \mathsf{m}_1)$ to its challenger. Let ct denote $\mathcal{C}$'s reply, $\mathcal{B}$ sends ct to $\mathcal{A}$.

In the subsequent query phase $\mathcal{B}$ behaves as described above. At the end of the experiment, $\mathcal{B}$ outputs the same bit b^* returned by $\mathcal{A}$ for the data-secrecy exper-

iment. Note that since $\mathcal{A}$ is given exactly the same challenge as in the semantic-security experiment, if $\mathcal{A}$ has a non-negligible advantage in breaking the data-secrecy of HIKE then $\mathcal{B}$ has the same non-negligible advantage in breaking the semantic security of LEEG, unless $\mathcal{B}$ aborts its simulation. Therefore we can conclude that: $\mathsf{Adv}^{\mathsf{sem.sec}}_{\mathsf{LEEG},\mathcal{B}}(\lambda) \geq \left(\frac{1}{Q_{\mathsf{id}}}\right)\mathsf{Adv}^{\mathsf{data.sec}}_{\mathsf{HIKE},\mathcal{A}}(\lambda)$. □

6.2 Token secrecy

Our notion of token-secrecy captures the idea that only the service provider $\mathcal{T}$ holding a valid decryption-token for a ciphertext that was created with $\mathcal{T}$ as intended recipient (*i.e.*with associated label of the form $\ell = (\cdot, \mathsf{pk}_{\mathcal{T}}, \cdot)$) can decrypt the message correctly. In other words, the adversary $\mathcal{A}$ (as a malicious $\mathcal{T}^*$) should not be able to decrypt the result of any computation $\mathcal{P}^*$ for which it did not received decryption-tokens. We recall that by the token-composability property (Remark 9 in Section 4), given two decryption-tokens $\mathsf{tok}_{\mathcal{P}}$, $\mathsf{tok}_{\mathcal{P}'}$ for two labelled programs $\mathcal{P}$ and $\mathcal{P}'$, it is possible to generate decryption tokens for any linear combination of the programs $\mathcal{P}$ and $\mathcal{P}'$.

$\mathsf{Exp}^{\mathsf{token.sec}}_{\mathsf{HIKE},\mathcal{A}}(\lambda)$:

$b \xleftarrow{\$} \{0,1\}, L_{\mathsf{tok}} = L_{\mathsf{lab}} = L_{\mathsf{keys}} = \emptyset, \Delta = \emptyset$
$\mathsf{pp} \leftarrow \mathsf{Initialise}(1^\lambda)$
$(\mathsf{id}^*, \mathsf{sk}_{\mathsf{id}^*}, \mathsf{pk}_{\mathsf{id}^*}) \leftarrow \mathcal{A}(\mathsf{pp})$
$L_{\mathsf{keys}} \leftarrow L_{\mathsf{keys}} \cup (\mathsf{id}^*, *, \mathsf{pk}_{\mathsf{id}^*})$
$O = \{O\mathsf{SignUp}(\cdot), O\mathsf{Encrypt}'(\cdot,\cdot), , O\mathsf{Disclose}(\cdot)\}$
$(\ell^*, \mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}^O(\mathsf{pp})$
let $\ell^* = (\mathsf{pk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}'}, \tau)$
if $\ell^* \notin L_{\mathsf{lab}}$ or $\ell^* \in L_{\mathsf{tok}}$ or $\mathsf{pk}_{\mathsf{id}} = \mathsf{pk}_{\mathsf{id}^*}$ or $\mathsf{pk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}'} \notin L_{\mathsf{keys}}$
  $\mathsf{ct} = \mathsf{error}$
else
  $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}_{\mathsf{id}}, \ell^*, \mathsf{m}_b)$
  $L_{\mathsf{tok}} \leftarrow L_{\mathsf{tok}} \cup \ell^*$, $L_{\mathsf{lab}} \leftarrow L_{\mathsf{lab}} \cup \ell^*$
$b^* \leftarrow \mathcal{A}^O(\mathsf{ct})$
if $b^* = b$ return 1, else return 0.

$O\mathsf{Encrypt}'(\ell, \mathsf{m})$:
parse $\ell = (\mathsf{pk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}'}, \tau)$
 if $\ell \in L_{\mathsf{lab}}$ or $\mathsf{pk}_{\mathsf{id}} = \mathsf{pk}_{\mathsf{id}^*}$ or $(\cdot,\cdot,\mathsf{pk}_{\mathsf{id}}) \notin L_{\mathsf{keys}}$
  return error.
$L_{\mathsf{lab}} \leftarrow L_{\mathsf{lab}} \cup \ell$
$\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}_{\mathsf{id}}, \ell, \mathsf{m})$
$\Delta \leftarrow \Delta \cup (\ell, \mathsf{ct})$
 return ct.

$O\mathsf{Disclose}(\mathcal{P})$:
parse $\mathcal{P} = (f, \ell_1, \ldots, \ell_n)$
with $\ell_i = (\mathsf{pk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}'}, \tau_i)$
 if $\mathsf{pk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}'} \notin L_{\mathsf{keys}}$ or $\mathsf{pk}_{\mathsf{id}} = \mathsf{pk}_{\mathsf{id}^*}$ or $\ell_i \in L_{\mathsf{tok}}$ for all $i \in [n]$
  return error.
$L_{\mathsf{tok}} \leftarrow L_{\mathsf{tok}} \cup \{\ell_1, \ldots, \ell_n\}$
$\mathsf{tok} = \mathsf{AllowAccess}(\mathsf{sk}_{\mathsf{id}}, \mathcal{P})$
 return tok.

Figure 22: The token-secrecy experiment and the oracles $O\mathsf{Encrypt}'$ and $O\mathsf{Disclose}$

In the token-secrecy experiment, we make use of the same $O\mathsf{SignUp}$ oracle as in the experiment in Figure 21; an $O\mathsf{Encrypt}'$ oracle which is the same as the $O\mathsf{Encrypt}$ in the experiment in Figure 21 except that every time it would output a ciphertext ct it will also add the record to the database, *i.e.* $\Delta \leftarrow \Delta \cup (\ell, \mathsf{ct})$ where ℓ is the label chosen by $\mathcal{A}$; and an additional $O\mathsf{Disclose}$ oracle, that enables $\mathcal{A}$ to get decryption-tokens of chosen (computations on) records. We allow the adversary to get decryption-tokens for any computation $\mathcal{P}$ as long as this does not contain the challenge labels.

Theorem 2. *The* HIKE *protocol achieves token-security,* i.e.*for any PPT adversary* $\mathcal{A}$ *taking part to the experiment in Figure 22, it holds that:*

$$\mathsf{Adv}^{\mathsf{token.sec}}_{\mathsf{HIKE},\mathcal{A}}(\lambda) = Pr\left[Exp^{\mathsf{token.sec}}_{\mathsf{HIKE},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \leq Q_{\mathsf{id}} \cdot \mathsf{Adv}^{\mathsf{sem.sec}}_{\mathsf{LEEG},\mathcal{A}}(\lambda).$$

Proof. We exhibit a reduction $\mathcal{B}$ that uses $\mathcal{A}$ to win the semantic-security experiment for the LEEG scheme. The reduction works exactly as the one in the proof of Theorem 1 a part for a couple of exceptions. First, this reduction holds an additional (private) list L_{rand}, that is empty at the beginning of the simulation. Second, $\mathcal{B}$ behaves differently (only) in the following cases:

Encryption queries: $\mathcal{B}$ forwards the queries to the $O\mathsf{Encrypt}'$ oracle, unless $\ell = (\mathsf{pk}^\star, \mathsf{pk}, \tau)$. In case $\ell = (\mathsf{pk}^\star, \mathsf{pk}, \tau)$, the reduction does not have the secret key for encryption and token generation. In order to simulate the encryption and be consistent with future token-generation queries, $\mathcal{B}$ checks if $(\ell, r) \in L_{\mathsf{rand}}$ for some value $r \in [0..q-1]$. If so, $\mathcal{B}$ uses the existing values r to compute the ciphertext $\mathsf{ct} = (\mathsf{m} \cdot P + r \cdot \mathsf{pk})$. Otherwise, $\mathcal{B}$ picks a random value $r \xleftarrow{\$} [0..q-1]$, updates $L_{\mathsf{rand}} \leftarrow L_{\mathsf{rand}} \cup (\ell, r)$, and computes $\mathsf{ct} = (\mathsf{m} \cdot P + r \cdot \mathsf{pk})$. In any case, $\mathcal{B}$ updates the list of queried labels $L_{\mathsf{lab}} \leftarrow L_{\mathsf{lab}} \cup \ell$, and returns ct to $\mathcal{A}$. Note that ct has the same distribution as the output of $\mathsf{Enc}(\mathsf{sk}^\star, \ell, \mathsf{m})$, indeed for any r chosen by the reduction there exists a value $r' \in [0..q-1]$ such that $r = r' \cdot \mathsf{sk}^\star \mod q$ and thus $\mathsf{ct} = \mathsf{m} \cdot P + r \cdot \mathsf{pk} = \mathsf{m} \cdot P + r' \cdot \mathsf{sk}^\star \cdot \mathsf{pk}$. The latter series of equalities shows that $\mathcal{B}$'s simulation is still perfect.

Disclose queries: $\mathcal{B}$ forwards to the $O\mathsf{Disclose}$ oracle all the queries $\mathcal{P} = (f, \ell_1 \ldots, \ell_n)$ with $f(x_1, \ldots, x_n) = a_0 + \sum_{i \in [n]} a_i x_i$, $\ell_i = (\mathsf{pk}, \mathsf{pk}', \tau_i)$ and $\mathsf{pk} \neq \mathsf{pk}^\star$. Otherwise, $\mathcal{P}$ contains labels of the form $\ell_i = (\mathsf{pk}^\star, \mathsf{pk}', \tau_i)$. The reduction performs the same checks as the $O\mathsf{Disclose}$ oracle, if any check fails $\mathcal{B}$ returns error. In case all conditions are met, $\mathcal{B}$ proceeds by checking if $(\ell_i, \cdot) \in L_{\mathsf{rand}}$ for all $i \in [n]$, in which case the reduction uses the randomness stored in L_{rand} to compute the token $\mathsf{tok} = (\sum_{i \in [n]} a_i r_i) \cdot \mathsf{pk}^\star$. Otherwise, for all those labels ℓ_j not present in L_{rand}, $\mathcal{B}$ samples a random element $r \xleftarrow{\$} [0..q-1]$ and updates the private list $L_{\mathsf{rand}} \leftarrow L_{\mathsf{rand}} \cup (\ell_j, r)$. At this point $(\ell_i, r_i) \in L_{\mathsf{rand}}$ for all the labels in the queried $\mathcal{P}$ and $\mathcal{B}$ can compute $\mathsf{tok} = (\sum_{i \in [n]} a_i r_i) \cdot \mathsf{pk}^\star$. In either case, $\mathcal{B}$ updates the list of queried token-labels, *i.e.* $L_{\mathsf{tok}} \leftarrow L_{\mathsf{tok}} \cup (\ell_1, \ldots, \ell_n)$, and returns tok to $\mathcal{A}$.

Let $(\ell^*, \mathsf{m}_0, \mathsf{m}_1)$ be $\mathcal{A}$'s input to the challenge phase. If $\ell^* \neq (\mathsf{pk}^\star, \cdot, \cdot)$ the reduction aborts ($\mathcal{A}$ chose to challenge a different client than the one $\mathcal{B}$ bet on). This event happens with probability $(1 - \frac{1}{Q_{\mathsf{id}}})$.

Otherwise $\ell^* = (\mathsf{pk}^\star, Q, \tau)$, $\mathcal{B}$ updates the list of queried labels $L_{\mathsf{lab}} \leftarrow L_{\mathsf{lab}} \cup \ell^*$ and sends $(\ell^*, \mathsf{m}_0, \mathsf{m}_1)$ to its challenger $\mathcal{C}$ for the semantic security game. Let ct denote $\mathcal{C}$'s reply, $\mathcal{B}$ forwards ct to $\mathcal{A}$.

In the subsequent query phase $\mathcal{B}$ behaves as described above.

At the end of the experiment, $\mathcal{B}$ outputs the same bit b^* returned by $\mathcal{A}$ for the token.sec experiment. Note that since $\mathcal{A}$ is given exactly the same challenge as in the sem.sec experiment, if $\mathcal{A}$ has a non-negligible advantage in winning the token.sec experiment, then $\mathcal{B}$ has the same non-negligible advantage in breaking the semantic-security of LEEG, unless $\mathcal{B}$ aborts its simulation. Therefore we conclude that:

$$\mathsf{Adv}^{\mathsf{sem.sec}}_{\mathsf{LEEG},\mathcal{B}}(\lambda) \geq \left(\frac{1}{Q_{\mathsf{id}}}\right) \cdot \mathsf{Adv}^{\mathsf{token.sec}}_{\mathsf{HIKE},\mathcal{A}}(\lambda)$$

□

6.3 Forgettability

Our notion of forgettability (forget.sec) captures the idea that after a forget request, the target ciphertext does no longer decrypt to the original message. More precisely, there is no way to derive what the original message was from a destroyed ciphertext.

Our forget.sec experiment, in Figure 23, uses the same oracles as the experiment in Figure 22. Concretely, Experiment 23 is like the token-secrecy experiments (Fig. 22) until the challenge phase. In this phase the forget.sec adversary challenges $\mathcal{C}$ with one

single new label ℓ. The challenger then randomly selects a message m, and encrypts it, generates the corresponding decryption token tok for $\mathcal{A}$, and runs the Forget procedure on the challenge ciphertext. Finally $\mathcal{C}$ returns to $\mathcal{A}$ the values $(\mathsf{ct}', \mathsf{tok})$. The adversary's goal is now to correctly guess the challenger's challenge message m. Let m^* denote the output of $\mathcal{A}$ at the end of the experiment in Figure 23, we say that $\mathcal{A}$ wins if $\mathsf{m}^* = \mathsf{m}$.

$\mathsf{Exp}^{\mathsf{forget.sec}}_{\mathsf{HIKE},\mathcal{A}}(\lambda)$:

$b \xleftarrow{\$} \{0,1\}, L_{\mathsf{tok}} = L_{\mathsf{lab}} = L_{\mathsf{keys}} = \emptyset, \Delta = \emptyset$
$\mathsf{pp} \leftarrow \mathsf{Initialise}(1^\lambda)$
$(\mathsf{id}^*, \mathsf{sk}_{\mathsf{id}^*}, \mathsf{pk}_{\mathsf{id}^*}) \leftarrow \mathcal{A}(\mathsf{pp})$
$L_{\mathsf{keys}} \leftarrow L_{\mathsf{keys}} \cup (\mathsf{id}^*, *, \mathsf{pk}_{\mathsf{id}^*})$

$O = \{O\mathsf{SignUp}(\cdot), O\mathsf{Encrypt}'(\cdot,\cdot),$
$\quad , O\mathsf{Disclose}(\cdot)\}$

$\ell^* \leftarrow \mathcal{A}^O(\mathsf{pp})$
parse $\ell^* = (\mathsf{pk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}'}, \tau)$

if $\ell^* \notin L_{\mathsf{lab}}$ or $\mathsf{pk}_{\mathsf{id}} = \mathsf{pk}_{\mathsf{id}^*}$ or $\mathsf{pk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}'} \notin L_{\mathsf{keys}}$
  $\mathsf{ct} = \mathsf{error}$
else
  $\mathsf{m} \xleftarrow{\$} \mathcal{M}$
  $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}_{\mathsf{id}}, \ell^*, \mathsf{m})$
  $\mathsf{ct}' \leftarrow \mathsf{Destroy}(\mathsf{ct})$
  $\Delta \leftarrow \Delta \cup (\ell^*, \mathsf{ct}')$
  $\mathsf{tok} \leftarrow \mathsf{AllowAccess}(\mathsf{sk}_{\mathsf{id}}, \mathcal{I}_{\ell^*})$
  $L_{\mathsf{tok}} \leftarrow L_{\mathsf{tok}} \cup \{\ell^*\}$; $L_{\mathsf{lab}} \leftarrow L_{\mathsf{lab}} \cup \{\ell^*\}$

$\mathsf{m}^* \leftarrow \mathcal{A}^O(\mathsf{ct}', \mathsf{tok})$
if $\mathsf{m}^* = \mathsf{m}$ return 1, else return 0.

Figure 23: The forgettability experiment

It is important to notice that the $\mathsf{forget.sec}$ experiment does not model an adversary that is able to obtain the original ciphertext ct, *e.g.* via a database backup or some previous random $\mathsf{retrieve}$ request. The main reason for this restriction is the necessity to deploy an access control system on the database Δ which is of independent interest. On the other hand, to avoid an old-ciphertext to be reused, we can only suggest that the client $\mathcal{U}$ never distributes the decryption token of queries that involve the label corresponding to *forgotten* ciphertexts.

In a nutshell, our forgettability security statement below says that after a forget request the user's record encrypts a random message. In particular, we are able to show that HIKE's Forget procedure achieves information theoretic security in 'hiding' the original message m even under the presence of a malicious server.[7]

Theorem 3. *The* HIKE *protocol achieves perfect forgettability,* i.e. *for any PPT adversary* $\mathcal{A}$ *taking part to the experiment in Figure 23, it holds that:*

$$Pr\left[Exp^{\mathsf{HIKE},\mathcal{A}}_{\mathsf{forget.sec}}(()\lambda) = 1\right] = \frac{1}{|\mathcal{M}|}$$

Proof. The result follows trivially from the information theoretic security of the $\mathsf{Destroy}$ algorithm demonstrated in Section 4. □

7 Implementation details and results

In this section, we discuss our *encoding map* from the message space to elliptic curve points. Afterwards, we describe the test-settings of our HIKE implementation with respect to different elliptic curve choices.

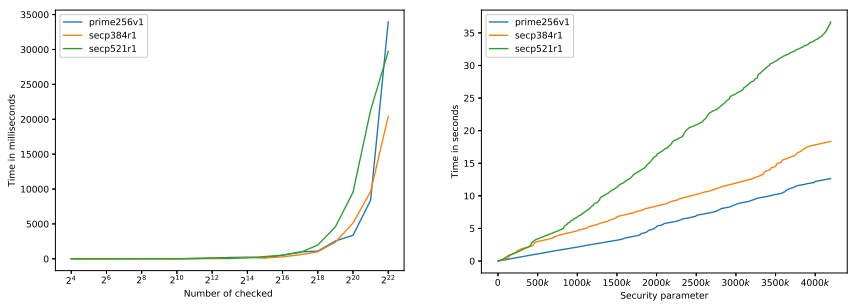
[7] More precisely, if the server is honest-but-curious except with forget requests.

Encoding Messages on the Elliptic Curve. A typical design problem that arises when using Elliptic Curve Cryptography (ECC) is to define an injective map ϕ from a message space $\mathcal{M}$ to the subgroup $\mathbb{G}$ generated by a point P on an elliptic curve $\mathcal{E}$. This problem was firstly considered and *"solved"* by Koblitz in [Kob87] by exploiting specific elliptic curves constructed over $\mathbb{F}_{2^n}$ for some appropriate n that depends on the message space dimension.

The main issue with Koblitz's map is that if we equip the message space $\mathcal{M}$ with an operation $\diamond$ and obtain the group $(\mathcal{M}, \diamond)$, then it is generally false that ϕ_K is a homomorphism between $(\mathcal{M}, \diamond)$ and $(\mathbb{G}, +)$, *i.e.*there exists two messages $\mathsf{m}_1, \mathsf{m}_2 \in \mathcal{M}$ such that $\phi_K(\mathsf{m}_1 \diamond \mathsf{m}_2) \neq \phi_K(\mathsf{m}_1) + \phi_K(\mathsf{m}_2)$. A more natural homomorphism map is given by $\phi : \mathbb{Z}_q \to \mathbb{G}$ as $\phi(\mathsf{m}) \coloneqq \mathsf{m} \cdot P$. The mapping is trivially a homomorphism when considering the message space as the natural group $(\mathbb{Z}_q, +)$. Unfortunately computing the inverse map ϕ^{-1} is exactly the DLog problem.

In our HIKE protocol we use this natural map to encode messages, and therefore the decryption procedure corresponds to solving an instance of the DLog problem. The apparent contradiction is addressed by the following observation. The security of HIKE relies on the hardness of solving the DLog problem (Assumption 1), but the efficiency of the decryption procedure is guaranteed by the feasibility of solving the IDLP in a particular interval (Assumption 2). In our implementation of HIKE, we consider the natural embedding ϕ and define a context-dependent message-space interval $\mathcal{M} = [a \dots b]$, for some $a, b \in \mathbb{N}$. This trick works whenever the decryption knows an approximation of the expected value. This is the case in most of the application scenarios we consider (*e.g.* range of blood pressure values and range of kilometres run per day). Additionally, the technique does not work when the message space is too big (*e.g.* floats of 64 bits) or not known.

To demonstrate our claim, we carried out one experiment to test that the decryption algorithm solves the IDLP in a reasonable time (see Figure 24b) whereas a malicious adversary would still face the full DLog problem which is infeasible (see Figure 24a).



(a) Time needed to solve the DLog.

(b) Time needed to solve IDLP.

Figure 24: Comparison between solving the DLog vs solving IDLP

We implemented an extremely naive brute-force attack that checks, sequentially and incrementally, all the points of the selected interval. For this algorithm the *worst-case* in the interval $[a, \dots, b]$ is the point $b \cdot P$. In Figure 24a, we empirically measure the running time of the our naive brute-force algorithm to solve the DLog problem with respect to the security parameter. As expected, the problem is exponentially hard. Then, in Figure 24b, we focus on a specific message space, *i.e.* numbers from 0 to 2^{22}

as justified by Assumption 2, and plot the required time needed to decrypt a specific message.

HIKE Implementation. We have developed our HIKE scheme on Python by creating a new cryptographic scheme in the Charm Crypto framework [AGM+13]. The source code of HIKE is freely available at https://github.com/Pica4x6/HIKE. For the experiments, we used a MacBook Air with 2,2 GHz Intel Core i7 and 8 GB of RAM. We executed the experiments 100 times independently using the timeit library and report the average of the execution times in Table 1.

	KGen	Enc	Dec	Eval
prime256v1	0.9ms	280.0ms	13442.4ms	20.2ms
secp384r1	1.0ms	399.7ms	15149.3ms	19.0ms
secp521r1	1.3ms	426.6ms	17102.7ms	189.2ms

	PublicKey	TokenGen	TokenDec	Destroy
prime256v1	5.3ms	1293.2ms	14.5ms	6.4ms
secp384r1	70.1ms	1521.0.0ms	86.7ms	73.9ms
secp521r1	66.4ms	1837.5ms	101.2ms	68.0ms

Table 1: Benchmark of HIKE scheme using the natural encoding map ϕ.

In addition, we evaluated the performances of HIKE using the three elliptic curves prime256v1, secp384r1 and secp521r1 that are recommended by the National Institute of Standard and Technology (NIST) [NIS17]. Note that our implementation is agnostic to the definition of elliptic curve, thus it can be easily adapted to work with any type of elliptic curves defined in [AGM+13].

We remark that for every experiment, we randomly select a message in the HIKE message space dimension in which IDLP is feasible by our Assumption 2 and our empirical test in Figure 24.

8 Conclusions and directions for future work

In this paper, we proposed a new labelled homomorphic encryption scheme for multivariate linear polynomial functions called LEEG. LEEG can be seen as a variant of ElGamal encryption on elliptic curve groups. We showed that LEEG supports additional features that are not commonly investigated for encryption scheme. We call this set of extra algorithms FEET, as it extends LEEG and improves its versatility. We then combined LEEG and FEET to make HIKE, a lightweight protocol designed for privately and securely store users' data while keeping it accessible to data owners and authorised service-providers. Application scenarios for HIKE include sport-tracking activity and simple e-Health alter systems. We deployed HIKE on Python and benchmarked its performance. Finally, we included in our security model some GDPR-inspired notions and proved that HIKE provides: *(i)* encrypted storage of the client's data; *(ii)* data owner's right to disclose information (including computation on data) to designated service-providers; and *(iii)* the right to be forgotten, *i.e.* the possibility for data owners to request that selected records be made un-recoverable.

We identify some direction for further development of our HIKE protocol. First, since HIKE is based on a semantic-secure homomorphic encryption scheme, it cannot tolerate a malicious server. It would be interesting to design protocols with no trust on the server, thus providing both data confidentiality and integrity. Second, there are other

extra features (not just FEET) that are worth developing. For example: generation of disclosure-tokens to allow any (chosen) third-party to decrypt a chosen computation on the user's data; introducing a trusted authority (*e.g.* a legal entity) with the power of decrypting malicious users' data only if it collaborates with the designated service providers; enabling secure *"editable decryption"* to support the *rectification right* (art. 16 in GDPR). Third, it would be worth investigating multi-key properties in LEEG. Such extension would for instance enable service-providers to perform statistic on data generated by different.

To the best of our knowledge, HIKE is the first cryptographic protocol proven to meet specific real-world privacy requirements, and we hope that it constitutes a springing-board for future works. We believe that a GDPR-oriented design of cryptographic protocols and primitives would facilitate developers implementation choices when designing new digital-services, as well as ensure cryptographically-proven security in the data-flow, leading to *privacy-by-design* solutions.

Acknowledgement. We thank the anonymous reviewers for their insightful comments and Erik-Oliver Blass for kindly shepherding us during this publication.

Lattice-Based Simulatable VRFs: Challenges and Future Directions

Carlo Brunetta, Bei Liang, and Aikaterini Mitrokotsa

Chalmers University of Technology, Gothenburg, Sweden

12th International Conference on Provable Security (PROVSEC), 2018
Presented at the 1st PROVSEC Workshop, Jeju (Rep. of Korea)

Journal of Internet Services and Information Security
Vol. 8, No. 4 (November, 2018)

Abstract: Lattice-based cryptography is evolving rapidly and is often employed to design cryptographic primitives that hold a great promise to be post-quantum resistant and can be employed in multiple application settings such as: e-cash, unique digital signatures, non-interactive lottery and others. In such application scenarios, a user is often required to prove non-interactively the correct computation of a pseudo-random function $F_k(x)$ without revealing the secret key k used. Commitment schemes are also useful in application settings requiring to commit to a chosen but secret value that could be revealed later.

In this short paper, we provide our insights on constructing a *lattice-based simulatable verifiable random function (sVRF)* using non interactive zero knowledge arguments and dual-mode commitment schemes and we point out the main challenges that need to be addressed in order to achieve it.

Keywords: Dual-Mode Commitment Scheme, Lattice-based Cryptography, Non Interactive Zero Knowledge Arguments, Pseudo Random Functions, Verifiable Random Functions

1 Introduction

Zero-knowledge (ZK) proofs [LLNW17] are employed to prove the knowledge of secret information while preserving the provers' privacy with respect to an NP language. Depending on whether the zero-knowledge proof is performed interactively or not, we may have *interactive* or *non-interactive* protocols; while the latter are more efficient regarding communication costs.

Pseudo-random functions (PRFs) [GGM86] are a very useful cryptographic primitive that is often employed in combination with *zero-knowledge* proofs in multiple application scenarios. Let us consider a general scenario: a prover $\mathcal{P}$ wants to prove to a verifier $\mathcal{V}$ the knowledge of a secret $\mathbf{w}$ and the correct computation of a PRF $F_{\mathbf{w}}$ on the input x, *i.e.,* $F_{\mathbf{w}}(x)$. A rather important question is:

How may $\mathcal{P}$ prove to $\mathcal{V}$ the correct evaluation of the PRF $F_{\mathbf{w}}(x)$ without leaking any information about $\mathbf{w}$, just by providing a proof π?

We consider the case where the communication between $\mathcal{P}$ and $\mathcal{V}$ should be **non**-interactive, *i.e.,* $\mathcal{P}$ needs to provide $\mathcal{V}$ all the necessary information to verify the correct computations in a single step.

The above stated question can be solved by employing a *verifiable random function* (VRF) [MVR99]. A VRF is a PRF with two additional algorithms; one algorithm that is able to generate a proof π of the correct computation of the PRF $F_{\mathbf{w}}(x)$ as well as a *verification* algorithm.

Verifiable random functions have a broad range of applications where the verification of a pseudorandom value is required. For instance, VRFs are employed in non-interactive lottery systems used in micropayments [MR02], domain name security extensions (DNSSEC) [GNP+15, PWH+17] as well as proof-of-stake blockchain protocols [LABK17, DGKR18]. For instance, recent papers [LABK17, DGKR18] use VRFs in *blockchain* consensus protocols *i.e.,* in order to either define a *fair and verifiable lottery* in which the winner will publish the next block, or as a way to generate a *"common and shared random string"* which can be seen as an equivalent of the CRS model.

Although algebraic pseudo-random functions and ZK proofs are well studied primitives, they have received limited attention in lattice settings; furthermore, to the best of our knowledge, *building lattice-based VRFs is an open problem.*

Lattice-based cryptographic primitives [Ajt96, Pei16], mainly rely on the *learning with errors* (LWE) [Reg10] and the *short integer solution* (SIS) [MP13] problems; they are quite promising for providing post-quantum resistance guarantees, while also offering simpler arithmetic operations and thus, important efficiency guarantees.

Designing a lattice-based VRF is a challenging and currently open problem since it requires a non-interactive proof in the standard model. As a step towards this direction, in this short paper, we provide our insights on designing a lattice-based *simulatable VRF* (sVRF), originally introduced by Chase and Lysyanskaya [CL07]. Informally, an sVRF is a VRF defined in a public parameter model, such as the *common random string* (CRS) model, which implies the existence of honest common parameters on the top of the standard VRF system. More precisely, besides the usual algorithms in a VRF there is an additional parameter generation algorithm which takes as input the security parameters and output the public parametrs pp. Both the input domain and the output range of the sVRF depend on pp. Meanwhile, pp is included in the inputs for all the algorithms KeyGen, Eval, Prove and Verify. Moreover, except of the uniqueness and pseudorandomness properties, sVRFs should also satisfy *simulatability* which is a novel property making them different from VRFs. Simulatability states that there exists a simulator that is able to simulate the common parameters such that, corresponding to a verification key, for any x, y, it is possible to produce a proof π that $F(sk, x) = y$. The simulated transcription is required to be indistinguishable from the values computed

from the parameters that are generated honestly. In this paper, we describe our insights on constructing an sVRF when relying on Libert *et al.*'s [LLNW17] method to prove zero-knowledge arguments for lattice-based PRFs. Furthermore, we describe the main challenges that need to be addressed in order to construct a lattice-based sVRF using this method.

1.1 A Roadmap to Lattice-based sVRFs

Chase and Lysyanskaya [CL07] provided a general construction of sVRFs from a perfectly binding computational hiding non-interactive commitment scheme and an unconditionally sound multi-theorem NIZK for NP. Their main idea is to use a multi-theorem NIZK to generate the proof for a statement that the public verification key is a commitment of the secret key and the function value is the correct result on the input applied to the secret-keyed PRF, *i.e.,* $\mathsf{pk} = \mathsf{Com}(\mathsf{sk}; r) \wedge y = F(\mathsf{sk}, x)$. However, such solution is based on a general assumption; in order to propose a lattice-based sVRF, we should specify a lattice-based PRF scheme.

Fortunately, thanks to the very recent significant results of Boneh *et al.* [BLMR13] who proposed a LWE-based key homomorphic PRFs as well as Libert *et al.*'s [LLNW17] three round zero-knowledge arguments of correct evaluation for the LWE-based PRF Boneh *et al.* [BLMR13] w.r.t. committed keys and inputs, it is possible to obtain a non-interactive solution of $y = F(\mathsf{sk}, x)$ as the correct evaluation of a PRF for a secret input x and a committed key sk. These results could be potentially employed in order to construct a lattice-based sVRF as we explore in this paper.

Libert *et al.* have significant contributions [LLNW17, LLM+16, LLNW18] in the area of designing efficient zero-knowledge proofs for lattice-related language. For instance, Libert *et al.* [LLM+16] considered how to construct zero-knowledge arguments of knowledge of a secret matrix X and vectors $\mathbf{s}, \mathbf{e}$ such that for a public vector $\mathbf{b}$ it holds $\mathbf{b} = \mathbf{X} \cdot \mathbf{s} + \mathbf{e} \bmod q$. Libert *et al.* [LLNW18] also investigated in the lattice setting how to design zero-knowledge arguments for the statement that c_x, c_y and c_z are the commitments to the polynomial-size bit-strings x, y and z which are the binary representations of large integers X, Y, Z satisfying certain algebraic relations such as $Z = X + Y$ and $Z = X \cdot Y$.

In order to obtain zero-knowledge arguments for the correct evaluation of the key-homomorphic PRF [8] proposed by Boneh *et al.* [BLMR13] , Libert *et al.* [LLNW17] presented a useful abstraction of Stern's protocol [Ste96] and they modified Boneh *et al.*'s lattice PRF [BLMR13] in order to efficiently prove the correct computation of the PRF value interactively, while providing zero-knowledge guarantees.

As stated in their paper [LLNW17], it is possible to obtain the first non-interactive lattice-based zero-knowledge protocol by directly applying the Fiat-Shamir transformation [FS87]. The main issue with this choice is that the Fiat-Shamir transformation is secure in the *Random Oracle Model* (ROM) which is against the original sVRF definition [CL07].

Thus, our main research objective is to find an appropriate transformation from ZK to NIZK, defined over lattices, not relying on the ROM. In Figure 25, we depict two different strategies in order to obtain a lattice-based sVRF: either by directly transforming Libert *et al.*'s ZK argument or by providing a different lattice-based ZK PRF proof system and applying a ZK to the NIZK transformation and then the Chase-Lysyanskaya's transformation from NIZK to sVRF.

[8] Namely demonstrating knowledge of a committed secret key $\mathbf{k}$, a secret input $\mathbf{J}$ and an output $\mathbf{y}$ satisfying $\mathbf{y} = F_{\mathbf{k}}(\mathbf{J})$

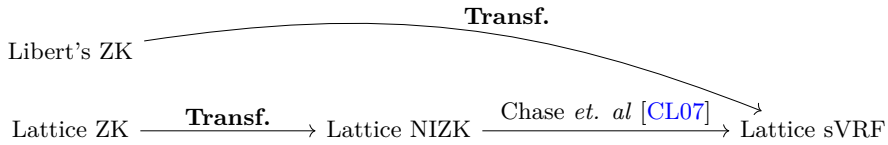


Figure 25: A roadmap to lattice-based sVRF. In **bold**, this paper's main research focus.

2 Applying Lindell's Tranformation

In this section, we provide our findings on defining an sVRF based on Libert's ZK argument [LLNW17] and Lindell's transformation [Lin15].

The latter [Lin15] can be applied to any sigma-protocol and transform it into a corresponding NIZK protocol. In contrast to Fiat-Shamir's transformation [FS87], Lindell's transformation does not require the random oracle model. More precisely, in Lindell's transformation the *zero-knowledge* property holds in the *common reference string* (CRS) model, while in order to achieve *soundness*, the used hash function is modeled as a *non-programmable* random oracle [Nie02]. In order to adopt Lindell's transformation an important requirement is that of a *dual-mode* commitment scheme.

The main concept of a commitment scheme is that it is possible to secretly fix some message m that is used in a protocol and in a second phase, open the commitment and therefore prove the correct knowledge or possession of the specific message m. Designing lattice-based commitment schemes has already received some attention in the literature [BKLP15, BDL$^+$16].

The *dual-mode* commitment represents the possibility to sample a statement in a language L via a bit b and use the commitment scheme in a *binding* way, *i.e.*, a commitment c can be decommitted in a *unique* message m, or in a *"trapdoor"* way, *i.e.,* that with some secret witness $\mathbf{w}$, it is possible to decommit c to any message m'.

Thus, the main property required for a dual-mode commitment scheme is that it is impossible to distinguish how the bit b is selected and therefore impossible to know if we are decommitting to the original message or we are using the trapdoor to decommit to an arbitrary message. A *dual-mode commitment scheme* represents a specific type of commitment schemes that are equivalently defined by Catalano and Visconti as *hybrid commitment schemes* [CV07].

As described in [Lin15], in order to define a dual-mode commitment scheme, Lindell requires a *membership-hard efficient-sampling language* defined as follows:

Definition 17 (Membership-hard with Efficient Sampling [Lin15])**.** *Let L be a language. L is* membership-hard with efficient sampling *(MHES) if there exists a probabilistic polynomial-time sampler S_L such that for every probabilistic polynomial-time distinguisher D there exists a negligible function $\mu(\cdot)$ such that:*

$$|Pr[D(S_L^x(0,1^n),1^n)=1]-Pr[D(S_L(1,1^n),1^n)=1]|\leq\mu(n)$$

where $S_L(b,\cdot)$ is a sampler that returns an instance in the language L if $b=0$ and an instance not in the language L if $b=1$. S_L^x denotes only the instance without the witness.

In a nutshell, the MHES language L is a language in which it is hard to distinguish if an efficient sampling algorithm S_L sampled the statement x in the language L or not: it is hard to decide the membership of $x \in L$ but it is easy to sample x in the language (or not).

In summary, in order to build an sVRF while employing the Lindell's transformation, the main building blocks required are depicted in Figure 26.

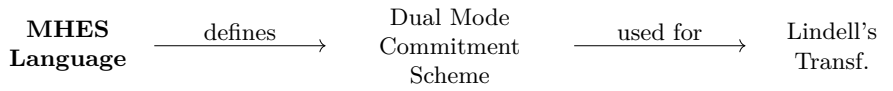


Figure 26: A roadmap to Lindell's transformation.

By assuming the hardness of the *inhomogeneous short integer solution* (ISIS) problem, if we follow the idea and structure of the DDH language construction proposed by Lindell [Lin15] in order to define the language L_{ls} of Eq. (11), the result is unfortunately not MHES for common lattice security parameters.

$$L_{\mathsf{ls}} \coloneqq \{(\mathbf{A}, \mathbf{B}, \mathbf{u}, \mathbf{v}) \mid \mathbf{A}, \mathbf{B} \in {\mathbb{Z}_p}^{n \times m}, \tilde{\mathbf{w}} \in \{0,1\}^m, \mathbf{u} = \mathbf{A}\tilde{\mathbf{w}}, \mathbf{v} = \mathbf{B}\tilde{\mathbf{w}}\}. \tag{11}$$

This is the case since whenever we provide a statement not in the language $(\mathbf{A}, \mathbf{B}, \mathbf{u}, \mathbf{v}) \notin L_{\mathsf{ls}}$, it exists in fact a statement $(\mathbf{A}, \mathbf{B}, \mathbf{A}\tilde{\mathbf{w}}', \mathbf{A}\tilde{\mathbf{w}}') \in L_{\mathsf{ls}}$ in the language for some $\tilde{\mathbf{w}}'$. Thus it cannot be used to define a dual-mode commitment scheme mainly because the commitment scheme will not be perfectly binding, which is a necessary condition in order to use Lindell's transformation.

If we do assume that there exists a hard problem in the format that given $\mathbf{A} \in \mathsf{Dom}$ and $\mathbf{y} \in \mathsf{Rang}$, it is hard to find $\mathbf{z} \in \mathsf{PreD}$ such that $\mathbf{Az} = \mathbf{y}$, then, on the ground of such an assumption, we can define our language as:

$$L \coloneqq \{(\mathbf{A}, \mathbf{B}, \mathbf{A}\tilde{\mathbf{w}}, \mathbf{B}\tilde{\mathbf{w}}) \mid \mathbf{A}, \mathbf{B} \in \mathsf{Dom}, \tilde{\mathbf{w}} \in \mathsf{PreD}\}. \tag{12}$$

The sampler S_L can be defined as: whenever $S_L(0, 1^n)$ outputs a random tuple $(\mathbf{A}, \mathbf{B}, \mathbf{A}\tilde{\mathbf{w}}, \mathbf{B}\tilde{\mathbf{w}})$, and $S_L(1, 1^n)$ outputs a random tuple $(\mathbf{A}, \mathbf{B}, \mathbf{A}\tilde{\mathbf{w}}, \mathbf{B}\tilde{\mathbf{w}}')$ of which $\mathbf{A}\tilde{\mathbf{w}} \neq \mathbf{A}\tilde{\mathbf{w}}'$ and $\mathbf{B}\tilde{\mathbf{w}} \neq \mathbf{B}\tilde{\mathbf{w}}'$. It is obvious to conclude that the language L defined as in Equation 12 is efficient sampling and membership-hard.

2.1 Dual-mode Commitment

In this section, we provide the formal definition of a dual-mode commitment scheme which is introduced by Lindell [Lin15] and present a dual-mode commitment scheme based on the language L.

Definition 18 (Dual-mode commitment scheme [Lin15])**.** *A dual-mode commitment scheme is a tuple of probabilistic polynomial-time algorithms (*GenCRS*,* Com*,* $\mathcal{S}_{\mathsf{com}}$*) such that:*

- $\mathsf{GenCRS}(1^\lambda)$: *outputs a common reference string, denoted* crs*,*
- $(\mathsf{GenCRS}, \mathsf{Com}, \mathsf{Decom}, \mathsf{ReceiverDecom})$: *When* $\mathsf{crs} \leftarrow \mathsf{GenCRS}(1^\lambda)$ *and* $m \in \{0,1\}^\lambda$*, the algorithm* $\mathsf{Com}_{\mathsf{crs}}(m; r)$ *with a random* r *is a non-interactive perfectly-binding commitment scheme with decommitment algorithm* Decom *and decommitment verification algorithm* $\mathsf{ReceiverDecom}$*.*
  (We require that $\mathsf{ReceiverDecom}_{\mathsf{crs}}(\mathsf{Com}_{\mathsf{crs}}(m; r), \mathsf{Decom}_{\mathsf{crs}}(m; r)) = m$ *except with negligible probability.)*
- $(\mathsf{Com}, \mathcal{S}_{\mathsf{com}})$: *For every probabilistic polynomial-time adversary* $\mathcal{A}$ *and every polynomial* $p(\cdot)$*, the output of the following two experiments are computationally indistinguishable.*

$\mathsf{Real}_{\mathsf{Com},\mathcal{A}}(1^\lambda)$:

(a) $\mathsf{crs} \leftarrow \mathsf{GenCRS}(1^\lambda)$; $\mathbf{c}, \mathbf{d} \leftarrow \emptyset$

(b) *For* $i = 1, \ldots, p(\lambda)$:

(a) $m_i \leftarrow \mathcal{A}(\mathsf{crs}, \mathbf{c}, \mathbf{d})$

(b) $c_i = \mathsf{Com}_{\mathsf{crs}}(m_i; r_i)$ *for* $r_i \in \{0,1\}^{\mathsf{poly}(\lambda)}$

(c) $d_i = \mathsf{Decom}_{\mathsf{crs}}(m_i; r_i)$

(d) *Set* $\mathbf{c} = c_1, \ldots, c_i$ *and* $\mathbf{d} = d_1, \ldots, d_i$

(c) *Output* $\mathcal{A}(\mathsf{crs}, m_1, \ldots, m_{p(\lambda)}, \mathbf{c}, \mathbf{d})$

$\mathsf{Simulation}_{\mathcal{S}_{\mathsf{com}}}(1^\lambda)$:

(a) $\mathsf{crs} \leftarrow \mathcal{S}_{\mathsf{com}}(1^\lambda)$; $\mathbf{c}, \mathbf{d} \leftarrow \emptyset$

(b) *For* $i = 1, \ldots, p(\lambda)$:

(a) $c_i \leftarrow \mathcal{S}_{\mathsf{com}}$

(b) $m_i \leftarrow \mathcal{A}(\mathsf{crs}, \mathbf{c}, \mathbf{d})$

(c) $d_i \leftarrow \mathcal{S}_{\mathsf{com}}(m_i)$

(d) *Set* $\mathbf{c} = c_1, \ldots, c_i$ *and* $\mathbf{d} = d_1, \ldots, d_i$

(c) *Output* $\mathcal{A}(\mathsf{crs}, m_1, \ldots, m_{p(\lambda)}, \mathbf{c}, \mathbf{d})$

Below we describe an instantiation of a dual-mode commitment scheme.

Protocol 1 (Instantiation of Dual-Mode Commitment)**.**

- Regular CRS generation: $\mathbf{A}, \mathbf{B} \leftarrow \mathit{Dom}$, $\tilde{\mathbf{w}}_1, \tilde{\mathbf{w}}_2 \leftarrow_R \mathit{PreD}$, *and compute* $\mathbf{w}_1 = \mathbf{A}\tilde{\mathbf{w}}_1$ *and* $\mathbf{w}_2 = \mathbf{B}\tilde{\mathbf{w}}_2$. *The* CRS *is* $(\mathbf{A}, \mathbf{B}, \mathbf{w}_1, \mathbf{w}_2)$.
- Alternative CRS generation (equivocal): *As above, except of the fact that we also choose a single* $\tilde{\mathbf{w}} \leftarrow_R \mathit{PreD}$ *and compute* $\mathbf{w}_1 = \mathbf{A}\tilde{\mathbf{w}}$ *and* $\mathbf{w}_2 = \mathbf{B}\tilde{\mathbf{w}}$.
- Commitment: *To commit to a bit* $e \in \{0,1\}$, *choose a random* $\mathbf{z} \leftarrow_R \mathit{PreD}$ *and compute* $\mathbf{y}_1 = \mathbf{A}\mathbf{z} - e\mathbf{w}_1$, $\mathbf{y}_2 = \mathbf{B}\mathbf{z} - e\mathbf{w}_2$. *The commitment is* $c = (\mathbf{y}_1, \mathbf{y}_2)$.
- Decommitment: *To decommit to* $c = (\mathbf{y}_1, \mathbf{y}_2)$, *provide* $(e, \mathbf{z})$.
- Receiver decommitment: *The receiver outputs* e *if* $\mathbf{A}\mathbf{z} = \mathbf{y}_1 + e\mathbf{w}_1$ *and* $\mathbf{B}\mathbf{z} = \mathbf{y}_2 + e\mathbf{w}_2$. *Otherwise, it outputs* $\perp$.
- Simulator $\mathcal{S}_{\mathsf{com}}$*:*
  (a) *Run the sampler* S_L *for the language* L *as equation (12) with input* $(0, 1^\lambda)$*:* i.e.,
  $$(\mathbf{A}, \mathbf{B}, \mathbf{A}\tilde{\mathbf{w}}, \mathbf{B}\tilde{\mathbf{w}}, \tilde{\mathbf{w}}) \leftarrow S_L(0, 1^\lambda)$$
  and set the CRS *as* $(\mathbf{A}, \mathbf{B}, \mathbf{A}\tilde{\mathbf{w}}, \mathbf{B}\tilde{\mathbf{w}})$. *Then,* $\mathcal{S}_{\mathsf{com}}$ *randomly samples* $\tilde{\mathbf{y}} \leftarrow_R \mathit{PreD}$ *and computes* $\mathbf{y}_1 = \mathbf{A}\tilde{\mathbf{y}}$ *and* $\mathbf{y}_2 = \mathbf{B}\tilde{\mathbf{y}}$. *Set* $c = (\mathbf{y}_1, \mathbf{y}_2)$.
  (b) *For a bit* $e \in \{0,1\}$, $\mathcal{S}_{\mathsf{com}}$ *computes* $\mathbf{z} = \tilde{\mathbf{y}} + e\tilde{\mathbf{w}}$, *and outputs the decommitment* $(e, \mathbf{z})$.

2.2 A Non-Interactive Zero-Knowledge Argument for a Lattice Based PRF

In this subsection, we provide a non-interactive zero-knowledge argument for the correct evaluation of the lattice-based PRF proposed by Boneh *et al.* [BLMR13]. We construct non-interactive zero-knowledge arguments of knowledge of a committed seed $\mathbf{k}$, a secret input $\mathbf{J}$ and an output $\mathbf{y}$ satisfying $\mathbf{y} = F_{\mathbf{k}}(\mathbf{J})$. We describe such arguments for the key-homomorphic PRF of Boneh *et al.* [BLMR13] and the PRF obtained by applying the Goldreich-Goldwasser-Micali (GGM) [GGM86] paradigm.

Recently, Libert *et al.* [LLNW17] have proposed zero-knowledge arguments for statements for which the given value $\mathbf{y} = \lfloor \prod_{i=1}^{L} \mathbf{P}_{\mathbf{J}[L+1-i]} \cdot \mathbf{k} \rfloor_p \in \mathbb{Z}_p^m$ is the correct evaluation for a committed seed $\mathbf{k} \in \mathbb{Z}_p^m$ and a secret input $\mathbf{J}[1] \dots \mathbf{J}[L] \in \{0,1\}^L$, where $\mathbf{P}_0, \mathbf{P}_1 \in \{0,1\}^{m \times m}$ are public binary matrices, without revealing neither $\mathbf{k}$ nor $\mathbf{J}[1] \dots \mathbf{J}[L]$. More precisely, they have used Stern's protocol [Ste96], which is adapted to handle correlated witnesses across relations modulo distinct integers.

An added ingredient to our recipe is Lindell's transformation [Lin15]: a Fiat-Shamir type transformation from sigma protocols to non-interactive zero knowledge argument; which employs a commitment scheme in the CRS model with the property that it is perfectly binding given the correctly constructed CRS, but it is equivocal to a simulator who generates the CRS in an alternative but indistinguishable way. In other words, the simulator can generate the CRS so that it looks like a real one, but a commitment can be decommitted to any value. In order to use Lindell's transformation in our context, a *lattice based dual-mode commitment scheme* is necessary.

In our Protocol 1, we show a concrete instantiation of a dual-mode commitment scheme. Following this, we can apply Lindell's transformation [Lin15] on Libert's abstract sigma-protocol [LLNW17]. In Libert's paper [LLNW17], there is the precise translation from the Boneh's PRF [BLMR13] to the abstract protocol's hypothesis which are extracted and summarized in Section 3. To avoid heavy notation, we will use just the general and abstract construction since the specific instantiation for the PRF is proved to be correct and implementable by Libert *et al.* [LLNW17].

Definition 19 (The abstract statement [LLNW17])**.** *Let $n_i, d_i \geq n_i$ be positive integers. Let $d = \sum_{i=1}^{N} d_i$. Suppose $\mathsf{VALID} \subseteq \{-1,0,1\}^d$ and $\mathcal{S}$ a finite set such that for any $\phi \in \mathcal{S}$ it is possible to associate to a permutation Γ_ϕ of d elements such that:*

$$\begin{cases} \mathbf{w} \in \mathsf{VALID} \iff \Gamma_\phi(\mathbf{w}) \in \mathsf{VALID} \\ \textit{If } \mathbf{w} \in \mathsf{VALID} \wedge \phi \textit{ uniform in } \mathcal{S} \implies \Gamma_\phi(\mathbf{w}) \textit{ uniform in } \mathsf{VALID} \end{cases}$$

Let us consider public matrices $\mathbf{M} := \{\mathbf{M_i} \in {\mathbb{Z}_{q_i}}^{n \times d_i}\}_{i \in [1..N]}$ and vectors $\mathbf{u} := \mathbf{u_i} \in \mathbb{Z}_{q_i}^{n_i}$, the prover argues in zero-knowledge the possession of integer vectors $\mathbf{w} := \{\mathbf{w} \in \{-1,0,1\}^{d_i}\}_{i \in [1..N]}$ such that:

$$\begin{cases} \mathbf{w} = (\mathbf{w_1} \| \mathbf{w_2} \| \dots \| \mathbf{w_N}) \in \mathsf{VALID} \\ \forall\, i \in [1..N] \,.\, \mathbf{M_i} \cdot \mathbf{w_i} = \mathbf{u_i} \pmod{q_i} \end{cases}$$

The described tuple $(\mathbf{M}, \mathbf{u})$ defines a statement of which $\mathbf{w}$ is the witness.

The protocol makes use of a statistically hiding and computationally binding string commitment scheme such as the SIS-based commitment of [KTX08]. Libert *et al.* have also shown that by assuming the commitment scheme $(\mathsf{Com}, \mathsf{Decom})$ to be a statistically hiding and computationally binding string commitment, then their protocol is a zero-knowledge argument of knowledge for the given statement with perfect completeness and soundness error 2/3. Based on the three round interaction protocol in [LLNW17] and Lindell's transformation [Lin15], by employing our lattice-based dual commitment scheme instantiated in Protocol 1, a non-interactive zero-knowledge argument for the correct evaluation of Boneh *et al.*'s lattice-based pseudo-random function [BLMR13] is yielded as follows:

Protocol 2. *Let Com be a statistically hiding and computationally binding string commitment scheme, for example the SIS-based commitment defined in [KTX08]. Let $(\mathsf{DRegularCRS}, \mathsf{DCom}, \mathsf{DReceiverDecom}, \mathsf{DDecom})$ be our lattice-based dual-mode commitment scheme of Protocol 1 and let $H_k : \{0,1\}^* \to \{1,2,3\}$ a keyed hash function.*

*Our **NIZK argument protocol*** $(\mathsf{GenCRS}, \mathsf{Prove}, \mathsf{Verify})$ *for the correct evaluation of Boneh* et al.*'s lattice-based PRF [BLMR13] is defined as the following three algorithms:*

- *Inputs: Let* $\mathbf{M} = \{\mathbf{M_i} \in \mathbb{Z}_{q_i}{}^{n_i \times d_i}\}_{i\in[N]}$ *be a set of matrices, for all* $i \in [1, \dots, N]$, *let* $\mathbf{w_i} \in \{-1, 0, 1\}^{d_i}$ *and* $\mathbf{w} = \|_{i=1}^{N} \mathbf{w_i} \in \mathsf{VALID}$.
  Let $\mathbf{u_i} := \mathbf{M_i} \cdot \mathbf{w_i} \pmod{q_i}$ *and define* $\mathbf{u} = \|_{i=1}^{N} \mathbf{u_i}$.
  The common input consists of $\mathbf{M} = \{\mathbf{M_i}\}_{i\in[N]}$ *and* $\mathbf{u} = \{\mathbf{u_i}\}_{i\in[N]}$, *while the prover's secret input (witness) is* $\mathbf{w} = \mathbf{w_1} \| \cdots \| \mathbf{w_N}$.

- GenCRS*: The* CRS *consists of the regular CRS* ρ *of our dual-mode commitment scheme, Protocol 1, and a key* s *for the hash function* H.

- Prove*: Takes as input the statement* $(\mathbf{M}, \mathbf{u})$, *the witness* $\mathbf{w}$ *and the CRS* ρ. *The algorithm computes three different commitments* C_1, C_2, C_3 *using the standard commitment scheme.*
  Afterwards, these commitments are then committed using the dual-mode commitment scheme and the commitment c *is computed and the decommit* (a, τ) *used for the decommitment. As a final step, depending on the hash of the statement and the commit* c, *we provide* b.

  The algorithm outputs the statement $(\mathbf{M}, \mathbf{u})$, *the commit* c, *the decommit information* (a, τ) *and* b. *We can see* (c, a, τ, b) *as the proof for* $(\mathbf{M}, \mathbf{u})$.

  Algorithm 1 describe it in details.

Algorithm 1 Prover $\mathcal{P}$: $\mathsf{Prove}((\mathbf{M}, \mathbf{u}), \mathbf{w}, \rho)$

(a) – Sample $\phi \leftarrow_R S$, for $i \in 1, \dots, N$ sample $\mathbf{r_i} \leftarrow_R \mathbb{Z}_{q_i}^{d_i}$ and define $\mathbf{r} = \|_{i=1}^{N} \mathbf{r_i}$ as the concatenation of $\mathbf{r_i}$s and $\mathbf{v} = \mathbf{w} \boxplus \mathbf{r}$ as $v_i = w_i + r_i \pmod{q_i}$ for all $i \in \{1, \dots, N\}$.
– Sample ρ_1, ρ_2, ρ_3 and compute

$$C_1 = \mathsf{Com}(\phi, \{\mathbf{M} \cdot \mathbf{r_i} \pmod{q_i}\}_{i=1}^{N}; \rho_1) \quad C_2 = \mathsf{Com}(\mathbf{\Gamma}_\phi(\mathbf{r}); \rho_2)$$

$$C_3 = \mathsf{Com}(\mathbf{\Gamma}_\phi(\mathbf{v}); \rho_3)$$

(b) Define $a = (C_1, C_2, C_3)$

(c) Compute $c = \mathsf{DCom}_\rho(a; \tau)$ and $\mathsf{DDecom}_\rho(a; \tau) = (a, \tau)$, where $\mathsf{DCom}_\rho(a; \tau)$ is our dual-mode commitment to a using randomness τ and CRS ρ, and (a, τ) is its corresponding decommitment;

(d) Compute $e = H_s((\mathbf{M}, \mathbf{u}), c)$

(e) Define b to be

$$b = \begin{cases} (\mathbf{\Gamma}_\phi(\mathbf{w}), \mathbf{\Gamma}_\phi(\mathbf{r}), \rho_2, \rho_3) & \text{when } e = 1 \\ (\phi, \mathbf{v}, \rho_1, \rho_3) & \text{when } e = 2 \\ (\phi, \mathbf{r}, \rho_1, \rho_2) & \text{when } e = 3 \end{cases}$$

Output: $\pi = ((\mathbf{M}, \mathbf{u}), c, a, \tau, b)$

- Verify*: Takes as input the statement* $(\mathbf{M}, \mathbf{u})$, *the commit* c, *the decommit information* a, τ *and* b.

Initially, the algorithm decommits c of the dual-mode commitment using (a, τ) in order to obtain $a = (C_1, C_2, C_3)$. Depending on the digest of the hash of the statement and c, a different check is made on C_i and b.

The output is 1 *if all the checks hold,* 0 *otherwise.*

Algorithm 2 describes the Verify *algorithm in details.*

Algorithm 2 Verifier $\mathcal{V}$: $\mathsf{Verify}((\mathbf{M}, \mathbf{u}), c, d, b)$

(a) Compute $a = \mathsf{DReceiverDecom}(c, (a, \tau))$. If $a = \perp$, output 0.

(b) Compute $e = H_s((\mathbf{M}, \mathbf{u}), c)$

(c) Compute and verify

  (a) If $e = 1$, let $b = (\mathbf{t}, \mathbf{s}, \rho_2, \rho_3)$. Check that $t \in \mathsf{VALID}$, $C_2 = \mathsf{Com}(s; \rho_2)$ and $C_3 = \mathsf{Com}(t \boxplus s; \rho_3)$.

  (b) If $e = 2$, let $b = (\pi, \mathbf{x}, \rho_1, \rho_3)$, parse $\mathbf{x} = (\mathbf{x_1} \| \cdots \| \mathbf{x_N}), \mathbf{x_i} \in \mathbb{Z}_{q_i}^{d_i}$, check that $C_2 = \mathsf{Com}(\pi, \{\mathbf{M_i} \cdot \mathbf{x_i} - \mathbf{u_i} \pmod{q_i}\}_{i=1}^{N}; \rho_1)$ and $C_3 = \mathsf{Com}(\mathbf{\Gamma}_\pi(\mathbf{x}); \rho_3)$.

  (c) If $e = 3$, let $b = (\psi, \mathbf{y}, \rho_1, \rho_2)$. parse $\mathbf{y} = (\mathbf{y_1} \| \cdots \| \mathbf{y_N}), \mathbf{y_i} \in \mathbb{Z}_{q_i}^{d_i}$ and check that $C_1 = \mathsf{Com}(\psi, \{\mathbf{M_i} \cdot \mathbf{y_i} \pmod{q_i}\}_{i=1}^{N}; \rho_1)$ and $C_2 = \mathsf{Com}(\mathbf{\Gamma}_\psi(\mathbf{y}); \rho_2)$.

(d) If the verification fails, output 0. Otherwise output 1.

3 Translation of Boneh's PRF

For completeness of the paper, we provide the specific instantiation of Boneh's lattice-based PRF [BLMR13] used in our Protocol 2, which is described and explained in details in Libert *et al.* [LLNW17].

For any t positive integer, define the following:

- $\mathbf{S}_t$: the set of all t-elements permutations.
- $\mathbf{B}_t^2$: the set of vectors in $\{0,1\}^{2t}$ with Hamming weight t.
- $\mathbf{B}_t^3$: the set of vectors in $\{-1,0,1\}^{3t}$ with exactly t elements equal to -1, t elements equal to 0 and t elements equal to 1.

Let Expand be the function that for every bit c and for all vectors $\mathbf{v} \in \mathbb{Z}^t$, it is defined as:

$$\mathsf{Expand}(c, \mathbf{v}) \coloneqq \begin{pmatrix} (1-c) \cdot \mathbf{v} \\ c \cdot \mathbf{v} \end{pmatrix} \in \mathbb{Z}^{2t}$$

Let $T_{c,\pi}$ be defined for every bit c, for all vectors $\mathbf{v} \coloneqq \begin{pmatrix} \mathbf{v_0} \\ \mathbf{v_1} \end{pmatrix} \in \mathbb{Z}^{2t}$ where $\mathbf{v_0}, \mathbf{v_1} \in \mathbb{Z}^t$ and for all permutation $\pi \in \mathbf{S}_t$, as

$$T_{c,\pi}(\mathbf{v}) \coloneqq \begin{pmatrix} \pi(\mathbf{v}_c) \\ \pi(\mathbf{v}_{1-c}) \end{pmatrix}$$

For any $B \in \mathbb{Z}, B > 0$, let us consider a specific way to represent integers, similar to the *binary representation of* B: define $\delta_B \coloneqq \lfloor \log_2 B \rfloor + 1$ and the sequence $\{B_j\}_{j \in [1..\delta_B]}$ with $B_j \coloneqq \left\lfloor \frac{B + 2^{j-1}}{2^j} \right\rfloor$ for every $j \in [1..\delta_B]$. For every integer $v \in [0..B]$, define $\mathsf{idec}_B(v) \coloneqq (v_{(1)}, v_{(2)}, \ldots, v_{(\delta_B)}) \in \{0,1\}^{\delta_B}$ such that $\sum_{j \in [1..\delta_B]} B_j v_{(j)} = v$.

Let $\sigma(x)$ be the standard sign function and define

$$\begin{gathered}\mathsf{vdec}'_{t,B} : [-B, B]^t \to \{-1, 0, 1\}^{t\delta_B} \\ (w_1, \ldots, w_t) \mapsto (\sigma(w_1) \cdot \mathsf{idec}_B(w_1), \ldots, \sigma(w_t) \cdot \mathsf{idec}_B(w_t))\end{gathered}$$

and $\mathsf{vdec}_{t,B} : [0, B]^t \to \{0, 1\}^{t\delta_B}$ as vdec' on the smaller domain $[0, B]^t$.
In order to close the change-representation function, let us define the matrix:

$$H_{t,B} = \begin{pmatrix} B_1 & \ldots & B_{\delta_B} & & & & & & \\ & & & B_1 & \ldots & B_{\delta_B} & & & \\ & & & & & & \ddots & & \\ & & & & & & & B_1 & \ldots & B_{\delta_B} \end{pmatrix} \in \mathbb{Z}^{t \times t\delta_B}$$

while, for all $\mathbf{v} \in [-B, B]^t$, it holds $H_{t,B} \cdot \mathsf{vdec}'_{t,B}(\mathbf{v}) = \mathbf{v}$
Next, let us consider specific maps that map a vector inside either $\mathbf{B}_t^2$ or $\mathbf{B}_t^3$: for all $\mathbf{v} \in \{0, 1\}^t$, define $\mathsf{TwoExt}(\mathbf{v}) \coloneqq (\mathbf{v} \| \mathbf{0}^{t-n_0} \| \mathbf{1}^{t-n_1}) \in \mathbf{B}_t^2$ where n_j is the number of coordinates in $\mathbf{v}$ equal to j. For all $\mathbf{v} \in \{-1, 0, 1\}^t$, define $\mathsf{ThreeExt}(\mathbf{v}) \coloneqq (\mathbf{v} \| \mathbf{0}^{t-n_0} \| \mathbf{1}^{t-n_1} \| (-\mathbf{1})^{t-n-1}) \in \mathbf{B}_t^3$ where n_j is the number of coordinates in $\mathbf{v}$ equal to j.

Similarly, for any $B \in \mathbb{Z}, B > 0$, let us consider the matrices:

$$\widehat{H}_{t,B} \coloneqq \left[\ H_{t,B} \ \middle|\ 0^{t \times t\delta_B}\ \right] \qquad \widetilde{H}_{t,B} \coloneqq \left[\ H_{t,B} \ \middle|\ 0^{t \times 2t\delta_B}\ \right]$$

that just extend the specific identity property of $H_{t,B}$ with respect to the image of TwoExt and $\mathsf{ThreeExt}$ respectively.

The next step is to transform Boneh's PRF [BLMR13]. Let us consider public binary matrices $P_0, P_1 \in \{0, 1\}^{m \times m}$, a committed seed $\mathbf{k} \in \mathbb{Z}_q^m$ and a secret bitstring $(J_1, \ldots, J_L) \in \{0, 1\}^L$ and define the matrix

$$P \coloneqq \left[\ P_0 \cdot \widehat{H}_{m,q-1} \ \middle|\ P_1 \cdot \widehat{H}_{m,q-1}\ \right] \in \mathbb{Z}_q^{m \times 4\overline{m}}$$

Let us consider public matrices $D_0 \in \mathbb{Z}_{q_1}^{n \times m_0}, D_1 \in \mathbb{Z}_{q_1}^{n \times \overline{m}}$ for some modulus q_1, for some integer m_0 and $\overline{m} = m\delta_{q-1}$. Let $\mathbf{r} \in [-\beta, \beta]^{m_0}$ be a discrete Gaussian vector with small β and define the vector

$$\widetilde{\mathbf{r}} \coloneqq \mathsf{ThreeExt}(\mathsf{vdec}'_{m_0,\beta}(r)) \in \mathbf{B}^3_{m_0\delta_\beta}$$

Define $\mathbf{x_0} = \mathbf{k}$, for each $i \in [1, L]$, compute $\mathbf{x_i} = P_{J_i}\mathbf{x_{i-1}} \pmod q$ and $\mathbf{y} = \lfloor \mathbf{x_L} \rfloor_p$. For all $i \in [1, L]$, define the vectors $\widehat{\mathbf{x}}_i \coloneqq \mathsf{TwoExt}(\mathsf{vdec}_{m,q-1}(\mathbf{x_i})) \in \mathbf{B}^2_{\overline{m}}$ and $\mathbf{s_{i-1}} \coloneqq \mathsf{Expand}(J_i, \widehat{\mathbf{x}}_{\mathbf{i-1}}) \in \mathbf{B}^2_{2\overline{m}}$.

Let $\widehat{z} \in \mathbf{B}^2_{\overline{m}}$. Let $\mathbf{w_1} \coloneqq (\widetilde{\mathbf{r}} \| \widehat{\mathbf{x}}_\mathbf{0})$, $\mathbf{M_1} = \left[\ D_0 \cdot \widetilde{H}_{m_0,\beta} \ \middle|\ D_1 \ \middle|\ \mathbf{0}^{n \times \overline{m}}\ \right]$ and $\mathbf{u_1} = \mathbf{M_1} \cdot \mathbf{w_1} \pmod{q_1}$.
Let $\mathbf{w_2} \coloneqq (\mathbf{s_0} \| \widehat{\mathbf{x}}_\mathbf{1} \| \mathbf{s_1} \| \widehat{\mathbf{x}}_\mathbf{2} \| \ldots \| \mathbf{s_{L-1}} \| \widehat{\mathbf{x}}_\mathbf{L})$, $\mathbf{u_2} = \mathbf{0}$ and

$$\mathbf{M_2} = \begin{bmatrix} P & -\widehat{H}_{m,q-1} & & \\ & \ddots & \ddots & \\ & & P & -\widehat{H}_{m,q-1} \end{bmatrix}$$

Let $\mathbf{w_3} \coloneqq (\widehat{\mathbf{x}}_\mathbf{L} \| \mathbf{z})$, $\mathbf{M_3} \coloneqq \left[\ (p \cdot \widehat{H}_{m,q-1}) \ \middle|\ \widehat{H}_{m,q-1}\ \right]$ and $\mathbf{u_3} \coloneqq q \cdot \mathbf{y}$.

Let us define $d_1 = 3m_0\delta_\beta + 2\overline{m}$, $d_2 = 6L\overline{m}$ and $d_3 = 4\overline{m}$ as the dimensions of $\mathbf{w_1}, \mathbf{w_2}$ and $\mathbf{w_3}$. Let $q_2 = q$, $q_3 = pq$ and $d = d_1 + d_2 + d_3$.

It holds $\mathbf{M_i} \cdot \mathbf{w_i} = \mathbf{u_i} \pmod{q_i}$ for $i \in \{1,2,3\}$ and $\mathbf{w} = (\mathbf{w_1} \| \mathbf{w_2} \| \mathbf{w_3}) \in \{-1,0,1\}^d$ of the form

$$\mathbf{w} = (\widetilde{\mathbf{r}} \| \widehat{\mathbf{x}}_\mathbf{0} \| \mathbf{s_0} \| \widehat{\mathbf{x}}_\mathbf{1} \| \mathbf{s_1} \| \widehat{\mathbf{x}}_\mathbf{2} \| \dots \| \mathbf{s_{L-1}} \| \widehat{\mathbf{x}}_\mathbf{L} \| \widehat{\mathbf{x}}_\mathbf{L} \| \mathbf{z})$$

Let us define the set VALID as the set of $\mathbf{w}$ such that:

- $\widetilde{\mathbf{r}} \in \mathbf{B}^3_{m_0 \delta_\beta}$ and $\widehat{\mathbf{x}}_\mathbf{0}, \dots, \widehat{\mathbf{x}}_\mathbf{L}, \mathbf{z} \in \mathbf{B}^2_{\overline{m}}$
- for all $i \in [1, L]$, $\mathbf{s_{i-1}} = \mathsf{Expand}(J_i, \widehat{\mathbf{x}}_\mathbf{i-1})$ for some $J_i \in \{0,1\}$

Let us define $\mathbf{S} \coloneqq \mathbf{S}_{3m_0\delta_\beta} \times (\mathbf{S}_{2\overline{m}})^{L+2} \times \{0,1\}^L$.
For every element $\pi = (\pi_r, \pi_0, \pi_1, \dots, \pi_L, \pi_z, b_1, \dots, b_L) \in \mathbf{S}$, let Γ_π be the permutation of the vector $\mathbf{w} \in \mathbb{Z}^d$ defined as

$$\begin{aligned}\Gamma_\pi(\mathbf{w}) \coloneqq (\pi_r(\widetilde{\mathbf{r}}) \| \pi_0(\widehat{\mathbf{x}}_\mathbf{0}) \| T_{b_1,\pi_0}(\mathbf{s_0}) \| \pi_1(\widehat{\mathbf{x}}_\mathbf{1}) \| T_{b_2,\pi_1}(\mathbf{s_1}) \| \\ \| \pi_2(\widehat{\mathbf{x}}_\mathbf{2}) \| \dots \| T_{b_L,\pi_{L-1}}(\mathbf{s_{L-1}}) \| \pi_L(\widehat{\mathbf{x}}_\mathbf{L}) \| \pi_L(\widehat{\mathbf{x}}_\mathbf{L}) \| \pi_L(\mathbf{z}))\end{aligned}$$

4 Challenges and Future Directions

In this section we will briefly discuss and collect our conjectures and/or our future research directions by dividing them into into two major classes: a first class of questions related to *transformations* from ZK to NIZK and a second class of challenges regarding *lattice languages*.

4.1 ZK Transformations

Choosing Lindell's transformation is not optimal for the final goal of constructing an sVRF since the transformation is defined in the non-programmable ROM.

Ciampi *et al.* [CPSV16] modified and improved Lindell's transformation: the transformation does not require the non-programmable random oracle *nor* a perfectly binding commitment scheme at the cost of a more specific language. By using Ciampi *et al.*'s transformation, it might be possible to obtain a ZK to NIZK transformation not based on the ROM.

Challenge 1. *Is it possible to use Ciampi* et al. *transformation in our sVRF construction-idea? The main challenge of this approach is to check if any lattice-based language can be defined in order to fulfil the transformation hypothesis and requirement.*

With the same spirit, we find an additional challenge of more general interest: a ZK to NIZK transformation that is not defined in the random oracle model (or any similar ones). Therefore, we state as a general challenge for future investigation:

Challenge 2. *Are there any other transformations in the literature that can be used for our construction-idea? Are they efficient? How do they compare among themselves or with respect to the Fiat-Shamir's transformation?*

4.2 Lattice Languages

When considering the Lindell's transformation, the language L_{IS} is ill-defined and therefore cannot be used in order to build a dual-mode commitment scheme. Furthermore, the language challenge of defining a membership-hard language can be seen as of perpendicular interest.

Challenge 3. *Is there a way to define a lattice-based membership-hard efficient sampling language L that can be used to define a dual-mode commitment scheme?*

Generally speaking and quite informally, the main obstacle is finding *"good"*-languages that have a *"unique-witness"*. This means that it would be incredibly useful to find a lattice-language L in which the witness of a statement $x \in L$ is unique. Solving this problem will open new directions in lattice-based cryptography.

Challenge 4. *Find a lattice-based language L in which every statement $x \in L$ has a unique witness w.*

As a different but related problem, if we consider a different ZK PRF proof system, the ZK language used for our construction-idea requires an additional property in order to be used by the Chase-Lysyanskaya's transformation. The ZK system has to be able to prove the correct computation of the PRF **and** the correctness of an additional commitment. It has to be defined over lattices **and**, after transforming it with the best ZK to NIZK transformation possible, the obtained NIZK has to be multi-theorem.

Challenge 5. *Given the best ZK transformation, find a ZK PRF argument/proof system that can be used for the Chase-Lysyanskaya's transformation.*

Acknowledgement.

We are grateful to the anonymous reviewers for their insightful comments, suggestions, discussions and the new literature-directions provided. This work was partially supported by the Swedish Research Council (Vetenskapsrådet) through the grant PRECIS (621-2014-4845).

Code-Based Zero Knowledge PRF Arguments

Carlo Brunetta, Bei Liang and Aikaterini Mitrokotsa

Chalmers University of Technology, Gothenburg, Sweden

22-th Information Security Conference (ISC) 2019
New York (USA)

Abstract: Pseudo-random functions are a useful cryptographic primitive that, can be combined with zero-knowledge proof systems in order to achieve privacy-preserving identification. Libert *et al.* (ASIACRYPT 2017) has investigated the problem of proving the correct evaluation of lattice-based PRFs based on the *Learning-With-Rounding* (LWR) problem. In this paper, we go beyond lattice-based assumptions and investigate, whether we can solve the question of proving the correct evaluation of PRFs based on code-based assumptions such as the *Syndrome Decoding* problem. The answer is affirmative and we achieve it by firstly introducing a very efficient code-based PRG based on the *Regular Syndrome Decoding* problem and subsequently, we give a direct construction of a code-based PRF. Thirdly, we provide a zero-knowledge protocol for the correct evaluation of a code-based PRF, which allows a prover to convince a verifier that a given output y is indeed computed from the code-based PRF with a secret key k on an input x, *i.e.,* $y = f(k, x)$. Finally, we analytically evaluate the protocol's communication costs.

Keywords: Coding Theory, Zero Knowledge, Pseudorandom Function, PRF Argument, Syndrome Decoding

1 Intro

Pseudo-random functions (PRFs) is a fundamental cryptographic primitive that can be employed to authenticate users, since they generate unique pseudorandom numbers. Zero-knowledge (ZK) proofs are often used to enforce honest behaviour or prove the identity of users, while providing strong privacy guarantees. By combining pseudo-random functions with zero-knowledge proofs, it is possible to achieve privacy-preserving user identification and answer the following question:

How may a prover $\mathcal{P}$ prove to a verifier $\mathcal{V}$, the correct evaluation of a PRF function $f(k, x) = y$, without leaking any information about k?

This is a rather important question with multiple applications, *e.g.,* e-cash, unique digital signatures, non-interactive lottery and more. Although algebraic (based on number-theoretic hardness assumptions) pseudo-random functions and zero-knowledge proofs, are well studied primitives; there has been comparatively *"less progress"* on these primitives based on post-quantum cryptographic assumptions such as code-based, hash-based, and multivariate-based.

Libert *et al.* [LLNW17] has recently addressed this problem based on lattice-based assumptions and more precisely, based on the *Learning-With-Rounding* (LWR) problem [BPR12] and provide a **lattice-based** zero-knowledge PRF argument.

Code-based cryptography enables the construction of cryptographic primitives that are believed to be secure against an adversary who has at his disposal a quantum computer. More precisely, code-based cryptographic primitives are based on assumptions related to the hardness of the *Syndrome Decoding* (SD) problem [BMv78], that has been proved to be NP-hard. Furthermore, except of their post-quantum nature, code-based cryptographic primitives offer significant advantages due to their significant algorithmic efficiency, offering several orders of complexity better than traditional cryptographic schemes.

In this paper, we focus on the construction of code-based cryptographic fundamental primitives, particularly on code-based pseudo-random generators/functions, as well as on code-based interactive zero-knowledge proof systems. We firstly introduce a code-based PRG and subsequently, we provide a direct construction of a code-based PRF. Finally, we provide a zero-knowledge protocol for the correct evaluation of the proposed code-based PRF and evaluate the protocol's communication cost.

Syndrome Decoding (SD). In this paper, we base our post-quantum cryptosystems on the hardness of the *Syndrome Decoding* (SD) problem [BMv78], which is a commonly used assumption in code-based cryptography. Recall that the SD problem with parameters n, r, ω is stated as follows: given a uniformly random matrix $\mathbf{H} \in \mathbb{F}_2^{r \times n}$ and a uniformly random syndrome $\mathbf{y} \in \mathbb{F}_2^r$, find a vector (word) $\mathbf{x} \in \mathbb{F}_2^n$ with Hamming weight ω, such that $\mathbf{H} \cdot \mathbf{x}^\intercal = \mathbf{y}^\intercal$. Berlekamp, McEliece and Tilborg [BMv78] proved that the SD problem is NP-complete, which implies that there is no polynomial-time algorithm for solving the SD problem in the worst case; however, many instances of the SD problem can be efficiently solved in the average case. Given existing results on the computing complexity for solving the SD problem (as reviewed by Chabaud and Stern [Cha95, Ste89]) it is the hardest to solve, when the weights of the words (*i.e.,* $\mathbf{x} \in \mathbb{F}_2^n$) are in the neighbourhood of the Gilbert-Varshamov bound [Gil52, Var57]. More precisely, we can set the weight of the words for an instance of the SD problem close to the Gilbert-Varshamov bound, such that the corresponding SD hardness assumption holds.

Considering the expensive computations required to transform binary strings into words of constant weight and length, the *Regular Syndrome Decoding* (RSD) [AFS05],

is a special case of the SD problem, where the words are restricted to *regular words*. Regular words are words of given weight w, that have a fixed number of 1's in each block of fixed size. The *Regular Syndrome Decoding* (RSD) problem is widely used in practical applications due to its high efficiency and convenience in generating words. For instance, Gaborit, Lauradoux, and Sendriern [GLS07] used regular words to improve Fischer and Stern's code-based PRG [FS96]. Let us consider binary words of length n and let us divide the coordinates in w blocks of n/w positions. A binary regular word of length n and weight w ((n, w)-regular word) has exactly one non-zero coordinate in each of these blocks. Notice that there is a reduction from the RSD to the SD problem, which implies that decoding a regular code cannot be more than about $\exp(w)$ easier than decoding a random code of the same weight.

Code-based Pseudo-random Generators/Functions. Fischer and Stern [FS96] proposed a simple and efficient construction of a pseudo-random generator (PRG), based on the intractability assumption for a special case of the SD problem, where $\mathbf{H} \in \mathbb{F}_2^{\lfloor \rho n \rfloor \times n}$, $\mathbf{x} \in \mathbb{F}_2^n$, $\omega = \lfloor \delta n \rfloor$ for some $\rho \in [0, 1]$ and $\delta \in [0, 1/2]$ such that the Gilbert-Warshamov bound denoted by $\mathsf{Bound}(\delta)$ satisfies the following condition: $\mathsf{Bound}(\delta) = -\delta \log_2 \delta - (1-\delta)\log_2(1-\delta) < \rho$. Thus, yielding a PRG $G_{\rho,\delta}(x) = \mathbf{H} \cdot \mathbf{x}^\intercal$ with domain $\mathbb{F}_2^n$ and range $\mathbb{F}_2^{\lfloor \rho n \rfloor}$. In order to obtain a PRG that outputs as many bits as we may want, Fischer and Stern [FS96] provided an iterative generator, which after computing $\mathbf{y} = \mathbf{H} \cdot \mathbf{x}^\intercal$, separates y as $\mathbf{y} = \mathbf{y}_1 \| \mathbf{y}_2$, where $\mathbf{y}_1$ denotes the first $\log_2 \binom{n}{\delta n}$ bits of $\mathbf{y}$ and $\mathbf{y}_2$ denotes the remaining bits. It outputs $\mathbf{y}_2$ and uses $\mathbf{y}_1$ as a new seed to compute $G_{\rho,\delta}$. We should note, that when performing this iteration, it is indispensable to have an efficient algorithm that computes a word with length n and weight $\omega = \lfloor \delta n \rfloor$ from a word of exactly $\log_2 \binom{n}{\omega}$ bits.

A pseudo-random function (PRF) is a function f_k with the property that no polynomial time attacker, when given oracle access to f_k, can distinguish f_k from a truly random function. Goldreich, Goldwasser, and Micali [GGM86] have shown how to generically construct a PRF from any length-doubling PRG (hence from any one-way function), known as the GGM paradigm, which requires n sequential invocations of the generator when operating on n-bit inputs. By plugging Fischer and Stern's code-based iterative PRG [FS96] into the sequential GGM paradigm [GGM86], we are able to obtain a code-based PRF. However, the PRF generated with this method is maximally sequential and very inefficient, since Fischer-Stern's PRG [FS96] uses a quadratic algorithm to transform binary strings of length $\log_2 \binom{n}{\omega}$ into words with length n and weight ω, while this algorithm has to be executed whenever the PRG evaluation is invoked in the GGM paradigm; thus, considerably slowing down the whole process. This motivates us to explore specialized constructions of PRFs under code-based assumptions that are much more efficient, than the previously described naive solution.

Zero-knowledge Proofs for the Correct computation of Code-based PRFs. Employing a PRF as a random oracle is limited to the setting where the *"key owner"*, *i.e.* the party that evaluates the PRF, should be fully trusted. Motivated by the fact that the key should remain private in this setting, we wish to establish a method that allows the owner of the key to prove to a verifier that the given value y is indeed the correct evaluation on an input point x, without revealing the key. Zero-knowledge (ZK) proof systems are very useful in numerous protocols, where a user has to prove knowledge of some secret information (*e.g.,* his identity), without revealing this information. Constructing a ZK protocol for the correct evaluation of a code-based PRF is quite challenging. There have been proposed ZK identification schemes [Ste96] based on the hardness of the SD problem and its variants [CVEYA11, AGS11], as well as identity-based identification schemes [CGG07, EYACM11]. There have also been proposed ZK proofs of plaintext knowledge based on the McEliece and the Niederreiter cryptosys-

tems [HMT13], as well as a ZK protocol in order to demonstrate that a given signature is generated by a certain certified user of a group, who honestly encrypts its identifying information [ELL+15]. Yet, we are not aware of any ZK protocol that can be employed to prove the correct evaluation of a code-based PRF.

Our Contribution. In this paper, we give a direct construction of PRF families based on coding theory, which is provably secure under code-based assumptions. More precisely, we take advantage of regular words, which can be very efficiently generated, and we build a new PRG by running two Fischer-Stern PRGs in parallel. Thus, avoiding the iteration needed in the Fischer-Stern PRG in order to output a bit string with doubled length. In this way, we obtain an efficient construction of PRF families from the *regular syndrome decoding* (RSD) problem [AFS05].

Secondly, we provide a zero-knowledge protocol for the correct evaluation of our code-based PRF, which allows a prover to convince a verifier that a given output $\mathbf{y}$ is indeed correctly computed from the code-based PRF with a secret key $\mathbf{k}$ held by the prover on the input $\mathbf{x}$. Such ZK protocols may be very useful in the context of oblivious PRF evaluations, which require the party who holds a PRF key to convince the other party that the key was correctly used in oblivious computations (*e.g.,* e-cash, unique digital signatures, non-interactive lottery). It is worth noting that, to the best of our knowledge, prior to our work there were few papers considering PRGs based on syndrome decoding [FS96, GLS07, MHC12] or other code-based assumptions [YS16], while no paper considers PRFs based on the SD assumption, let alone considering the problem of proving the correct evaluation of a code-based PRF. We believe that our results would certainly help to bring more interest into code-based cryptography and enhance its important roles in the post-quantum cryptography era.

Overview of Our Techniques. Let us consider an (n, w)-regular word of length n and weight w. We divide the coordinates in w blocks of n/w positions, and a (n, w)-regular word has exactly one non-zero coordinate in each of these blocks. If n and w are chosen such that $n/w = 2^b$, then there is a mapping $\phi_{n,w}$ from $\mathbb{F}_2^{wb}$ to the (n, w)-regular words in $\mathbb{F}_2^n$.

Let $\mathbf{H}_0, \mathbf{H}_1 \in \mathbb{F}_2^{r\times n}$ where $r = w \cdot b$ and $n = w \cdot 2^b$ and $f : \mathbb{F}_2^r \to \mathbb{F}_2^{2r}$ as:

$$f(\mathbf{k}) = \begin{pmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{pmatrix} \cdot \phi(\mathbf{k})^\intercal = \begin{pmatrix}\mathbf{H}_0 \cdot \phi(\mathbf{k})^\intercal\\ \mathbf{H}_1 \cdot \phi(\mathbf{k})^\intercal\end{pmatrix} = (\mathbf{y}_0, \mathbf{y}_1)^\intercal$$

For an input bit string $\mathbf{x} \in \mathbb{F}_2^t$ and by applying the GGM paradigm, we can therefore define a code-based PRF as follows $\mathsf{PRF} : \mathbb{F}_2^r \times \mathbb{F}_2^t \to \mathbb{F}_2^r$, where:

$$\mathsf{PRF}_{\mathbf{k}}(\mathbf{x}) = \mathsf{PRF}_{\mathbf{k}}\big((x_1, \cdots, x_t)\big) = f_{x_t}(f_{x_{t-1}}(\cdots(f_{x_1}(\mathbf{k}))\cdots).$$

The pseudo-randomness of our code-based PRF could be reduced to the hardness of the underlying *regular syndrome decoding* (RSD) problem and the unpredictability of the Goldreich-Levin hardcore bit, similarly to [MHC12].

Let us now explain the core idea of how we may build a zero-knowledge protocol for the correct evaluation of our proposed code-based PRF, which allows a prover to convince a verifier that a given output $\mathbf{y}$ is correctly computed from the PRF using a secret key $\mathbf{k}$ on input $\mathbf{x}$, namely $\mathbf{y} = f_{x_t}(f_{x_{t-1}}(\cdots(f_{x_1}(\mathbf{k}))\cdots)$. Without loss of generality, let us consider the case for input length of $t = 2$. Given $\mathbf{x} = (x_1, x_2)$, according to our PRF construction, it holds:

$$\begin{pmatrix}\mathbf{H}_{x_1} & 0\\ 0 & \mathbf{H}_{x_2}\end{pmatrix}\begin{pmatrix}\phi(\mathbf{k})^\intercal\\ \phi(f_{x_1}(\mathbf{k}))^\intercal\end{pmatrix} = \begin{pmatrix}f_{x_1}(\mathbf{k})^\intercal\\ f_{x_2}(f_{x_1}(\mathbf{k}))^\intercal\end{pmatrix}$$

If we reveal all the intermediate results, *i.e.,* the value $\mathbf{y}^1 = f_{x_1}(\mathbf{k})$ which is exactly the seed used to compute the next GGM iteration *i.e.,* the value $\mathbf{y} = \mathbf{y}^2 = f_{x_2}(\mathbf{y}^1)$,

then it is possible for a malicious verifier to compute the PRF on different inputs $\mathbf{x}' = (x_1, 1 - x_2)$ (without knowing the secret key $\mathbf{k}$), which subsequently could be used to break the pseudo-randomness of the PRF. Therefore, we have to *"hide"* the intermediate evaluations while proving the correctness of the PRF evaluation. This goal is accomplished by introducing a specific map ϕ^{-1} that can be used to hide all the intermediate evaluation results while maintaining the Stern's protocol format. Formally, we obtain:

$$\begin{pmatrix} \mathbf{H}_{x_1} & \phi^{-1} \\ 0 & \mathbf{H}_{x_2} \end{pmatrix} \cdot \begin{pmatrix} \phi(\mathbf{k})^\intercal \\ \phi(\mathbf{y}^1)^\intercal \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{y}^\intercal \end{pmatrix}$$

By embedding the above technique into Stern's ZK protocol framework [FS96], we obtain an interactive ZK argument system, in which, given the input and output values $\mathbf{x}, \mathbf{y}$,the prover is able to prove that $\mathbf{y} = \mathsf{PRF}_{\mathbf{k}}(\mathbf{x})$ is indeed the evaluation of $f_{x_t}(f_{x_{t-1}}(\cdots(f_{x_1}(\mathbf{k}))\cdots)$. The protocol is repeated many times to achieve negligible soundness error.

Related Work. Libert *et al.* [LLNW17] have investigated the problem of correctly evaluating arguments for lattice-based pseudo-random functions *w.r.t.* committed keys and inputs, using (interactive) zero-knowledge proofs; this is achieved by providing an abstraction of Stern's protocol [Ste96] based on lattices. Brunetta *et al.* [BLM18] further investigated the possibility of using Libert *et al.*'s results in order to construct more advanced primitives such as *simulatable verifiable random functions* (sVRF). However the following question is left open:

"Is it possible to achieve a ZK PRF argument based on other (non lattice-based) post-quantum assumptions?"

Motivated and inspired by these works, we show that it is indeed possible to construct PRF families based on coding theory assumptions and that it is possible to use the original Stern's protocol to achieve the ZK argument.

Goldreich-Goldwasser-Micali Construction. In 1986, Goldreich, Goldwasser and Micali [GGM86] proposed a generic transformation from any PRGs that doubles the input length, into a family of PRFs. This elegant construction is the main core of our PRF and the reason of our main interest in code-based PRGs.

Code-based PRGs and Stream Ciphers. In 1996, Fischer-Stern [FS96] defined a simple PRG based on the *syndrome decoding* (SD) problem. A decade later, Gaborit *et al.* published a code-based stream cipher called SYND [GLS07], which is an improvement of Fischer-Stern's PRG, revisited as a stream-cipher. Meziani *et al.* proposed 2SC [MCEYA11], a code-based sponge-function stream cipher. Shortly after, Meziani *et al.* improved the SYND cipher and defined X-SYND [MHC12], which is a stream-cipher based on the *regular syndrome decoding* (RSD) problem and of which we get inspiration for our constructions.

Stern's Protocol. In 1996, Stern [Ste96] published a code-based identification protocol with a zero-knowledge property. Different improved versions are defined by Aguilar *et al.* [AGS11] or Cayrel *et al.* [CVEYA11] in order to reduce the soundness error. In our constructions, we have employed the original zero-knowledge identification protocol proposed by Stern [Ste96] , given the simplicity of the construction and its generality.

Paper organisation. In Section 2, the paper notation and the minimal coding-theory background is reported. In Section 3, we present our code-based PRG construction and by applying the GGM transformation, we obtain our code-based PRF. In Section 4, we

describe our PRF proof argument that is compatible with the Stern's protocol statements and, by applying Stern's protocol, we achieve a code-based ZK PRF argument. In Section 5, we describe an application scenario for our protocol and we discuss the protocol's communication cost. Finally, in Section 6, we summarize our results and point out to possible future directions.

2 Preliminaries

This section provides the minimal coding theory definitions needed and the notation used in the paper. We will recall some coding hard problems and we will conclude the section by reporting Stern's zero-knowledge identification protocol [Ste96].

Let $\mathbb{N}$ be the set of positive integers and let the uniform sampling of x in a set X defined as $x \leftarrow_R X$. Let us denote with $|x|$ the length of the bit-representation of x. We denote with $\mathsf{I2B}_b(n)$ the map that takes an integer value n and outputs the b-bit binary representation $\mathbf{x} \in \mathbb{F}_2^b$. We denote with $\mathsf{B2I}(\mathbf{x})$ the map that takes a binary string $\mathbf{x}$ and outputs the corresponding integer value n.

A linear code $\mathcal{C}$ of an n-dimensional vector space over a finite field $\mathbb{F}_q$ is a k dimensional subspace where q is a prime power, k and n are integers and $0 < k < n$. The elements $\mathbf{y} \in \mathbb{F}_q^n$ are called *words* and, if they are part of the code, *i.e.* $\mathbf{y} \in \mathcal{C}$, then, they are called *codewords*. The weight of a word $\mathbf{x}$ is denoted as $\mathsf{wt}(\mathbf{x})$ and it counts the number of non-zero components of the word $\mathbf{x}$. A code $\mathcal{C}$ can be represented by a generator matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ as $\mathcal{C} = \{\mathbf{x} \cdot \mathbf{G} \mid \mathbf{x} \in \mathbb{F}_q^k\}$, where k is the number of rows and n the number of columns and the multiplication $\cdot$ is the standard matrix multiplication.

Given the vector subspace description of the code $\mathcal{C}$, the dual-code $\mathcal{C}^\perp$ is generated by a parity check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$. For the matrix $\mathbf{H}$, it holds $\mathcal{C} = \{\mathbf{x} \in \mathbb{F}_q^n \mid \mathbf{H} \cdot \mathbf{x}^\intercal = 0\}$. Throughout the paper, we will consider only binary codes, *i.e.* $q = 2$, and therefore, we use $\oplus$ to represent the bit-wise XOR operation.

Let us consider the integers $n, k, r \in \mathbb{N}$ and the parity check matrix $\mathbf{H} \in \mathbb{F}_2^{r \times n}$ of the code $\mathcal{C}$ of dimension k over $\mathbb{F}_2^n$, in which we consider $r = n - k$.

Assumption 3 (Binary Syndrome Decoding (SD))**.** *Given a binary matrix $\mathbf{H} \in \mathbb{F}_2^{r \times n}$, a binary vector $\mathbf{y} \in \mathbb{F}_2^r$ and an integer $w > 0$, find a **word** $\mathbf{x} \in \mathbb{F}_2^n$ such that $\mathsf{wt}(\mathbf{x}) = w$ and $\mathbf{H} \cdot \mathbf{x}^\intercal = \mathbf{y}$.*

The SD problem is known to be NP-complete [BMv78]. We are interested in a simplified version of the SD problem in which the word $\mathbf{x}$ is **regular**, *i.e.* for $\mathbf{x}$ with weight w, it can be split into w equal-blocks of length $\frac{n}{w}$ and each of them has a single non-zero entry.

Assumption 4 (Regular Syndrome Decoding ($\mathsf{RSD}(n, r, w)$))**.** *Given a binary matrix $\mathbf{H} \in \mathbb{F}_2^{r \times n}$, a binary vector $\mathbf{y} \in \mathbb{F}_2^r$ and an integer $w > 0$, find a **regular word** $\mathbf{x} \in \mathbb{F}_2^n$ such that $\mathsf{wt}(\mathbf{x}) = w$ and $\mathbf{H} \cdot \mathbf{x}^\intercal = \mathbf{y}$.*

Augot *et al.* [AFS05] prove the NP-completeness of the RSD problem and we will base the security of our constructions on this specific problem.

Stern's protocol [Ste96] is a zero-knowledge sigma-protocol that describes the language L defined as the elements $(\mathbf{M}, \mathbf{y}) \in \mathbb{F}_2^{r \times n} \times \mathbb{F}_2^r$ of which there exists a witness $\mathbf{s} \in \mathbb{F}_2^n$, such that $\mathsf{wt}(\mathbf{s}) = w$ and $\mathbf{M} \cdot \mathbf{s} = \mathbf{y}$. Stern's protocol requires a commitment scheme Com and allows a prover $\mathcal{P}$ to prove to a verifier $\mathcal{V}$ the knowledge of the witness vector $\mathbf{s}$ given the statement $(\mathbf{M}, \mathbf{y})$.

Theorem 4 (Stern's protocol)**.** *From the original paper [Ste96], Stern's protocol, as reported in Figure 27, is correct, has soundness probability of $\frac{2}{3}$ and it is zero-knowledge.*

Statement: $(\mathbf{M}, \mathbf{y}) \in \mathbb{F}_2^{r \times n} \times \mathbb{F}_2^r$ and witness $\mathbf{s} \in \mathbb{F}_2^n$ with $\mathsf{wt}(\mathbf{s}) = w$ and $\mathbf{M} \cdot \mathbf{s} = \mathbf{y}$.

1. **Commitment**: $\mathcal{P}$ samples $\mathbf{r} \leftarrow_R {\mathbb{F}_2}^n$, a permutation π of the set $\{1, \dots, n\}$ and random values ρ_0, ρ_1, ρ_2 for the commitment scheme Com. The result of applying the permutation π to a vector $\mathbf{r} = (r_1, \dots, r_n)$ is $\pi(\mathbf{r}) = (r_{\pi(1)}, \dots, r_{\pi(n)})$.
   The prover $\mathcal{P}$ sends the following commitments to the verifier $\mathcal{V}$:

$$C_0 = \mathsf{Com}((\pi, \mathbf{M} \cdot \mathbf{r}^\intercal)\,; \rho_0) \quad C_1 = \mathsf{Com}(\pi(\mathbf{r})\,; \rho_1) \quad C_2 = \mathsf{Com}(\pi(\mathbf{r} \oplus \mathbf{s})\,; \rho_2)$$

2. **Challenge**: $\mathcal{V}$ sends the challenge $c \in \{0, 1, 2\}$ to $\mathcal{P}$
3. **Response**: $\mathcal{P}$ sends to $\mathcal{V}$, based on the challenge c, the reply: *if* $c = 0$, $(\pi(\mathbf{r}), \pi(\mathbf{s}))$ and ρ_1, ρ_2; *if* $c = 1$, $(\mathbf{r} \oplus \mathbf{s}, \pi)$ and ρ_0, ρ_2; and *if* $c = 2$, $(\mathbf{r}, \pi)$ and ρ_0, ρ_1.

Verification: given the challenge c and the response $(\tilde{\mathbf{r}}, \tilde{\pi})$, $\mathcal{V}$ verifies:

if $c = 0$: given ρ_1, ρ_2, $\mathcal{V}$ checks $\mathsf{wt}(\tilde{\pi}) \stackrel{?}{=} w$, $\mathsf{Com}(\tilde{\mathbf{r}}\,; \rho_1) \stackrel{?}{=} C_1$ and $\mathsf{Com}(\tilde{\mathbf{r}} \oplus \tilde{\pi}\,; \rho_2) \stackrel{?}{=} C_2$

if $c = 1$: given ρ_0, ρ_2, $\mathcal{V}$ checks $\mathsf{Com}((\tilde{\pi}, \mathbf{M} \cdot \tilde{\mathbf{r}}^\intercal \oplus \mathbf{y}^\intercal; \rho_0) \stackrel{?}{=} C_0$ and $\mathsf{Com}(\tilde{\pi}(\tilde{\mathbf{r}}); \rho_2) \stackrel{?}{=} C_2$

if $c = 2$: given ρ_0, ρ_1, $\mathcal{V}$ checks $\mathsf{Com}((\tilde{\pi}, \mathbf{M} \cdot \tilde{\mathbf{r}}^\intercal\,; \rho_0) \stackrel{?}{=} C_0$ and $\mathsf{Com}(\tilde{\pi}(\tilde{\mathbf{r}})\,; \rho_1) \stackrel{?}{=} C_1$

In each case, $\mathcal{V}$ outputs 1 if and only if all the checks are correct.

Figure 27: Stern's protocol description.

Let π a permutation of the set $\{1, \dots, n\}$ if we assume that $|\pi| > n$, and $|\mathsf{Com}|$ is the commitment length, then the communication cost of the protocol is:

$$\mathsf{Cost}_{\mathsf{Stern}}(n, r) \leq \Big(\underbrace{3 \cdot |\mathsf{Com}|}_{\text{Commitment}} + \overbrace{2}^{\text{Challenge}} + \underbrace{n + |\pi| + |\rho_0| + \max_{i \in \{1,2\}} |\rho_i|}_{\text{Response}} \Big) \text{ bits}$$

3 Code-Based PRF

In this section, inspired by Gaborit's [GLS07] and Meziani's [MHC12] code-based stream ciphers, we define our own simple PRG G that has double-length pseudorandom output. Furthermore, after proving that f is indeed a PRG, we present our code-based PRF obtained by employing the Goldreich-Goldwasser-Micali (GGM) transformation [GGM86].

Let $w, b \in \mathbb{N}$ positive integers chosen such that $n = 2^b w$, $r = w \cdot b$, and the related RSD problem $\mathsf{RSD}(n, r, w)$ of Assumption 4 is hard. Consider the binary words $\mathbf{s} \in \mathbb{F}_2^n$ of length n and composed by w blocks of length 2^b, *i.e.* $\mathbf{s} = (\mathbf{s}_1, \dots, \mathbf{s}_w)$ such that every block $\mathbf{s}_j$ has weigth $\mathsf{wt}(\mathbf{s}_j) = 1$. We are interested in maps that have binary regular words as image.

Let us define the map ϕ as the map that takes a bit-string $\mathbf{y} \in \mathbb{F}_2^r$ and outputs a regular word $\mathbf{s} \in \mathbb{F}_2^n$ such that $\mathsf{wt}(\mathbf{s}) = w$ and that is computed as follows.

Firstly, the binary string $\mathbf{y}$ is divided into w blocks as $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_w)$ of which each block $\mathbf{y}_i$ is a binary string with length b. Then, for every $j \in \{1, \dots, t\}$, we compute the integer value n_j represented by the block $\mathbf{y}_j$ and denote it as $\mathsf{B2I}(\mathbf{y}_j) = n_j$. In this way, we transform the vector $(\mathbf{y}_1, \dots, \mathbf{y}_w)$ into a vector of integers $(n_1, \cdots, n_w)$, where every n_j is contained in the interval $\{0, \dots, 2^b - 1\}$. Since there are 2^b possible values for n_j, we bijectively identify every integer with a canonical vector of length 2^b. This bijection takes as input an integer n_j and outputs the canonical vector $\mathbf{e}_{n_j+1} \in \mathbb{F}_2^{2^b}$, which is the binary vector of length 2^b, with a single 1 in position $n_j + 1$.

Finally, we transform the integer vector and obtain a vector of canonical vectors $(\mathbf{e}_{n_1+1}, \cdots, \mathbf{e}_{n_w+1})$ that are concatenated and output by ϕ. In summary, the map ϕ is computed as:

$$\phi(\mathbf{y}) = \phi\big((\mathbf{y}_1, \dots, \mathbf{y}_w)\big) = \big(\mathbf{e}_{\mathsf{B2I}(\mathbf{y}_1)+1} \| \cdots \| \mathbf{e}_{\mathsf{B2I}(\mathbf{y}_w)+1}\big) = \mathbf{s}$$

It is trivial to observe that $\mathbf{s}$ is a regular word of length n and weigth w since $\mathbf{s}$ is the concatenation of w canonical vectors of length 2^b and the weight $\mathsf{wt}(\mathbf{s})$ is equivalent to the sum of the weight of the canonical vectors, which is w. It is important to note, that ϕ can be efficiently computed and therefore, we assume that the computational cost is constant.

For example, the vector $\mathbf{y} = \big(01\|11\|00\big)$ would be transformed into the regular word $\phi(\mathbf{y}) = \mathbf{s} = \big(\mathbf{e}_2\|\mathbf{e}_4\|\mathbf{e}_1\big) = \big(0100\|0001\|1000\big)$.

After defining the map ϕ, we are interested in developing a pseudorandom generator (PRG) based on the RSD assumption (see Assumption 4), inspired by Meziani's [MHC12] stream-cipher design. Let us first report both definitions.

Definition 20 (Pseudorandom Generator (PRG) [KL08])**.** *Given the positive integers $\ell_{\mathsf{in}}, \ell_{\mathsf{out}} \in \mathbb{N}$ with $\ell_{\mathsf{out}} > \ell_{\mathsf{in}}$, let $G : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}$ be a deterministic function. We say that G is a **pseudorandom generator** if the following two distributions are computationally indistinguishable:*

- *Sample a random seed $s \in \{0,1\}^{\ell_{\mathsf{in}}}$ and output $G(s)$.*
- *Sample a random string $r \in \{0,1\}^{\ell_{\mathsf{out}}}$ and output r.*

A *stream cipher* is an encryption scheme used in contexts where the messages are *streams*, *i.e.* the messages do not have a fixed-length a priori, and therefore a key-*"stream"* has to be generated and used. In order to do so, stream-ciphers are usually designed with an *initialization* algorithm that takes a key and initialize the cipher into an **internal state**. Consecutively, the cipher has an *output* algorithm that outputs a fixed-length key-stream based on the internal state and an *update* algorithm that *"evolves"* the internal state.

As described also by Fischer-Stern [FS96], it is natural to build stream-ciphers from PRGs: the stream cipher key is indeed the initial PRG's seed s. Then, we can compute $G(s)$ and use the first ℓ_{in} bits as the *internal state* and the remaining $\ell_{\mathsf{out}} - \ell_{\mathsf{in}}$ as the key-stream output. By iterating the PRG computation using the always different internal state, we obtain an arbitrary long key-stream.

Given the strong connection between stream ciphers and PRGs, we focus on Meziani *et al.*'s [MHC12] code-based stream cipher, depicted in Figure 28.

Definition 21 (X-SYND Stream Cipher [MHC12])**.** *Let $w, b \in \mathbb{N}$ be positive integers and define $n = w2^b$, $r = wb$. Let $\mathbf{A}_0, \mathbf{A}_1 \leftarrow_R \mathbb{F}_2^{r \times n}$ be random binary matrices. Define the **X-SYND stream cipher** as:*

- *$\mathsf{Init}(\mathsf{IV}, \mathbf{s})$: given an initialization vector IV and a seed $\mathbf{s}$ both of length $\frac{r}{2}$, let $\mathbf{z}^\intercal = \mathbf{A}_0 \cdot \phi\big((\mathbf{s}\|\mathsf{IV})\big)^\intercal \oplus (\mathbf{s}\|\mathsf{IV})^\intercal$ and set as initial state $\mathsf{st}_0{}^\intercal = \mathbf{A}_1 \cdot \phi(\mathbf{z})^\intercal \oplus \mathbf{z}^\intercal$;*
- *$\mathsf{Upd}(\mathsf{st}_i)$: given the internal state st_i, update the state $\mathsf{st}_{i+1}{}^\intercal = \mathbf{A}_0 \cdot \phi(\mathsf{st}_i)^\intercal$;*
- *$\mathsf{Out}(\mathsf{st}_i)$: given the state st_i, output the key-stream $\mathsf{sk}_{i+1}{}^\intercal = \mathbf{A}_1 \cdot \phi(\mathsf{st}_i)^\intercal$*

Similarly to X-SYND, let $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{F}_2^{r \times n}$, where $r = w \cdot b$ and $n = w \cdot 2^b$ and the map $\phi : \mathbb{F}_2^r \to \mathbb{F}_2^n$ as before. Let us define the function $f : \mathbb{F}_2^r \to \mathbb{F}_2^{2r}$ as:

$$f(\mathbf{k}) = \begin{pmatrix}\mathbf{A}_0 \\ \mathbf{A}_1\end{pmatrix} \cdot \phi(\mathbf{k})^\intercal = \begin{pmatrix}\mathbf{A}_0 \cdot \phi(\mathbf{k})^\intercal \\ \mathbf{A}_1 \cdot \phi(\mathbf{k})^\intercal\end{pmatrix} = \begin{pmatrix}\mathbf{y}_0 \\ \mathbf{y}_1\end{pmatrix} \tag{13}$$

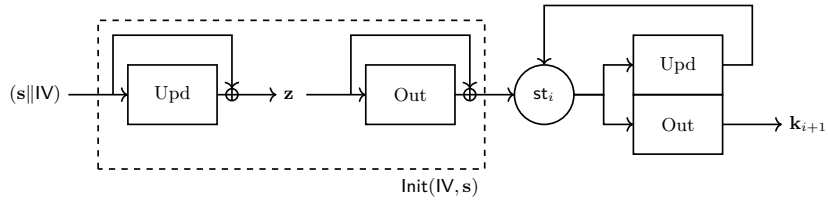


Figure 28: A high-level representation of the X-SYND stream cipher.

It has to be observed that f is indeed Meziani *et al.*'s [MHC12] computation of the initialization and update algorithms, *i.e.* $f(\mathsf{st_i})^\intercal = (\mathsf{Upd}(\mathsf{st}_i)^\intercal \| \mathsf{Out}(\mathsf{st}_i)^\intercal)$.

This observation allows us to reuse Meziani *et al.* X-SYND proofs and easily prove that f is indeed a PRG.

Proposition 6. *f is a PRG that reduce to a $\mathsf{RSD}(n, 2r, w)$ problem (Assum. 4).*

Proof. We sketch the main idea of the proof in two parts, that follow the same reasoning as Meziani *et al.*'s [MHC12] X-SYND's security proof parts.

We start by observing that since ϕ is a bijection between vectors in $\mathbb{F}_2^r$ and regular words in $\mathbb{F}_2^n$ with weight w, it is obvious that knowing a regular word solution $\mathbf{x}$ is equivalent of knowing the vector $\mathbf{k}$ such that $\phi(\mathbf{k}) = \mathbf{x}$.

Given this observation, in fact, we have an $\mathsf{RSD}(n, 2r, w)$ instance since:

$$f(\mathbf{k})^\intercal = \begin{pmatrix}\mathbf{A}_0\\ \mathbf{A}_1\end{pmatrix} \cdot \phi(\mathbf{k})^\intercal = \begin{pmatrix}\mathbf{A}_0\\ \mathbf{A}_1\end{pmatrix} \cdot \mathbf{x}^\intercal \in \mathsf{RSD}(n, 2r, w)$$

The second step is pseudorandomness and to prove it, we use the fact that Meziani *et al.* prove in Theorem 2 [MHC12], using Goldreich-Levin hard-core bit theorem [GL89], that the map $(\mathsf{Upd}, \mathsf{Out})$ is a PRG. Given the observation that $(\mathsf{Upd}, \mathsf{Out})$ is exactly f, we can conclude that f is indeed a PRG. □

From PRG to PRF. Finally, we use our PRG f and construct a PRF.
In order to do so, we use the Goldreich-Goldwasser-Micali construction [GGM86]. For the sake of clarity, let us report the PRF definition and the GGM construction.

Definition 22 (Pseudorandom Function (PRF))**.** *Let S be a distribution over $\{0,1\}^\ell$ and $F_s : \{0,1\}^m \to \{0,1\}^n$ be a family of functions indexed by strings s in the support of S.*

We say $\{F_s\}$ is a pseudorandom function family if for every p.p.t. adversary D, there exists a negligible function ϵ such that:

$$|\Pr[D^{F_s}(\cdot) = 1] - \Pr[D^R(\cdot) = 1]| \leq \epsilon,$$

where s is distributed according to S, and R is a function sampled uniformly at random from the set of all functions from $\{0,1\}^m$ to $\{0,1\}^n$.

Definition 23 (GGM Construction [GGM86])**.** *Let $G : \{0,1\}^\ell \to \{0,1\}^{2\ell}$ be a length-doubling PRG and $s \in \{0,1\}^\ell$ be a seed for G. Write $G(s) = (G_0(s), G_1(s))$ with G_0,$G_1 : \{0,1\}^\ell \to \{0,1\}^\ell$. Then, on input $\mathbf{x} \in \{0,1\}^m$, we define the **GGM pseudorandom function** $F_s : \{0,1\}^m \to \{0,1\}^n$ as*

$$F_s(\mathbf{x}) = F_s((x_1, \ldots, x_\ell)) = G_{x_\ell}(G_{x_{\ell-1}}(\cdots(G_{x_1}(s))\cdots) \tag{14}$$

Theorem 5. *If $G : \{0,1\}^\ell \to \{0,1\}^{2\ell}$ is a PRG, then $\{F_s\}$ is a PRF family.*

Having fixed a positive non-null integer $t \in \mathbb{N}$, let us define our PRF $\mathsf{PRF} : \mathbb{F}_2^r \times \mathbb{F}_2^t \to \mathbb{F}_2^r$ as the PRF obtained by transforming our PRG f of Eq. (13) with the GGM construction of Eq. (14). For readability, we will always denote the key in subscript, *i.e.* $\mathsf{PRF}(\mathbf{k}, \mathbf{x}) = \mathsf{PRF}_{\mathbf{k}}(\mathbf{x})$. Formally we have,

$$\mathsf{PRF}_{\mathbf{k}}(\mathbf{x}) = \mathsf{PRF}_{\mathbf{k}}\big((x_1, \cdots, x_t)\big) = f_{x_t}\Big(f_{x_{t-1}}\Big(\cdots\big(f_{x_1}(\mathbf{k})\big)\Big)\cdots\Big) \tag{15}$$

Corollary 1. *By Theorem 5,* PRF *is a code-based PRF.*

4 Code-Based Zero Knowledge PRF Argument

In this section, we describe how our PRF construction can be adapted to be compatible with Stern's protocol [Ste96] and thus, achieve a Zero-Knowledge (ZK) PRF argument, *i.e.* can be employed to prove the correctness of a PRF evaluation. We will start from a naïve description of a Stern-like statement and explain a specific security-flow that seems not to be easily solvable. To solve the problem, we define the map ψ that will act as the inverse map ϕ^{-1} and modify accordingly the statement in order to obtain a secure Stern-like statement $\big((\mathbf{M}, \mathbf{y}), \mathbf{s}\big)$.

Briefly, Stern's protocol allows a prover $\mathcal{P}$ to prove the knowledge of a witness $\mathbf{s}$ with weight w to a verifier $\mathcal{V}$, that holds a public statement $\big(\mathbf{M}, \mathbf{y}\big)$. The statement and the witness are related to the equation $\mathbf{M} \cdot \mathbf{s}^\intercal = \mathbf{y}^\intercal$.

At a first glance, we can observe that our PRG G is already defined in a Stern-like format **but** iterating the PRG requires the application of the map ϕ, which has no possible linear representation. Let us consider the GGM iterative structure and let $\mathbf{y}^i$ be the i-th partial evaluation of the PRF, while x_{i+1} be the next *"branching"* in the GGM construction. The $(i+1)$-th partial evaluation is computed as $\mathbf{A}_{x_{(i+1)}} \cdot \phi(\mathbf{y}^i)^\intercal = \mathbf{y}^{(i+1)\intercal}$. Since ϕ has no-linear representation, it is indeed impossible to re-write the equation as a single matrix $\overline{\mathbf{M}} \in \mathbb{F}_2^{r \times n}$ that multiplies only the secret initial vector $\phi(\mathbf{k})$.

It is important to note that, in order to use Stern's protocol, the witness is required to have a specific weight w and therefore we will consider as witness the regular word $\phi(\mathbf{k})$. This observation allows us to rewrite the PRF evaluation as a system of equations that describe all the singular partial evaluations and can be directly used to run Stern's protocol. Let $\mathbf{k} = \mathbf{y}^0$ and $\mathbf{x} = (x_1, \ldots, x_t)$ and $\mathbf{y}^t = \mathsf{PRF}(\mathbf{k}, \mathbf{x})$. Then, it formally holds that:

$$\begin{cases} \mathbf{A}_{x_1} \cdot \phi(\mathbf{y}^0)^\intercal = \mathbf{y}^{1\intercal} \\ \mathbf{A}_{x_2} \cdot \phi(\mathbf{y}^1)^\intercal = \mathbf{y}^{2\intercal} \\ \quad \ldots \\ \mathbf{A}_{x_t} \cdot \phi(\mathbf{y}^{t-1})^\intercal = \mathbf{y}^{t\intercal} \end{cases} \iff \begin{pmatrix} \mathbf{A}_{x_1} & 0 & & \\ 0 & \mathbf{A}_{x_2} & 0 & \\ & \ddots & \ddots & \ddots \\ & & 0 & \mathbf{A}_{x_t} \end{pmatrix} \cdot \begin{pmatrix} \phi(\mathbf{y}^0)^\intercal \\ \phi(\mathbf{y}^1)^\intercal \\ \vdots \\ \phi(\mathbf{y}^{t-1})^\intercal \end{pmatrix} = \begin{pmatrix} \mathbf{y}^{1\intercal} \\ \mathbf{y}^{2\intercal} \\ \vdots \\ \mathbf{y}^{t\intercal} \end{pmatrix} \tag{16}$$

Unfortunately, this representation has a security-flaw that allows a malicious adversary $\mathcal{A}$ to compute the PRF on different inputs $\mathbf{x}'$ without requiring the knowledge of $\mathbf{k}$. This flaw is not captured by the GGM transformation and Stern's protocol security model, since the problem is related to the *"composition"* of the construction and the unusual behaviour observed when naïvely merging the security models. We call this composed-protocol as **prove-on-demand** protocol, in which we can either solely compute the PRF and, in a different moment in time, request to execute the ZK arguments. Further discussion is presented in Section 5.

In a nutshell, let $\mathcal{A}$ be an adversary whose goal is to distinguish between our code-based PRF PRF and a random function ζ, *i.e.*, break the pseudo/randomness property and related security model. Whenever the adversary queries a value $\mathbf{x}$, $\mathcal{A}$ can either ask to obtain just the value $\mathsf{PRF}(\mathbf{k}, \mathbf{x})$ or to obtain the transcript of the execution of Stern's protocol, which contains $\mathsf{PRF}(\mathbf{k}, \mathbf{x})$ too. It is trivial to notice that the challenger can reply to the second query type by applying the simulatable property of ZK protocols, *i.e.*, providing a simulated transcript that correctly verifies Equation (16) and obtains a random value by evaluating ζ.

On the other hand, this naïve ZK proof gives access to $\mathcal{A}$ to **all** the partial evaluations of the GGM transformation. $\mathcal{A}$ can take, *w.l.o.g.*, the partial evaluation $\mathbf{y}^{t-1}$ and correctly compute $\mathbf{A}_{1-x_t} \cdot \phi(\mathbf{y}^{t-1})^\intercal = \mathsf{PRF}\big(\mathbf{k}, \mathbf{x}'\big)$, which is a valid PRF evaluation of the input $\mathbf{x}'$ with a different t-th component. With this knowledge, $\mathcal{A}$ can query the challenger on $\mathbf{x}'$ and just verify if the answer is equivalent to its computation or not, therefore distinguishing between PRF and ζ. *Mutatis mutandis*, $\mathcal{A}$ can personally compute any input $\mathbf{x}'$ except the ones that have a different first input-bit x_1. This is because $\mathcal{A}$ does not hold the pre-computation $\mathbf{y}^0$, which is exactly the secret key $\mathbf{k}$.

Similarly, it is possible to find other uncommon attacks that break other security properties, *e.g.*, the soundness property for Stern's protocol. The reason of all these problems is the disclosure of the partial evaluations, that completely break the GGM transformation Theorem 5 proof. For this reason, our goal is to *"hide"* the partial evaluation, while maintaining the simple and elegant representation compliant with Stern's protocol statement.

Let us consider the map $\psi : \mathbb{F}_2^n \to \mathbb{F}_2^r$ that takes a regular word $\mathbf{w}$ of weight w and outputs a binary vector of length r. The main design property of ψ is to invert the map ϕ and to be representable in a linear matrix format.

First of all, let $\mathbf{w}$ be a regular word of length $n = w \cdot 2^b$ that representa $\mathbf{w}$ as the concatenation of w canonical vectors, *i.e.* $\mathbf{w} = \big(\mathbf{e}_{n_1} \mid \cdots \mid \mathbf{e}_{n_w}\big)$. Let $\mathsf{I2B}_b$ be the map that given an integer j, it outputs, as a row vector, the binary representation in b-bit, *i.e.*, zeros are added accordingly if necessary.

Let us consider the binary matrix ψ as:

$$\psi = \left(\underbrace{\left(\mathsf{I2B}_b(0)^\intercal \| \ldots \| \mathsf{I2B}_b\left(2^b-1\right)^\intercal\right) \| \ \cdots \ \| \left(\mathsf{I2B}_b(0)^\intercal \| \ldots \| \mathsf{I2B}_b\left(2^b-1\right)^\intercal\right)}_{w \text{ times}} \right) \tag{17}$$

By notation abuse, let the evaluation of the map ψ be the matrix multiplication with the matrix ψ in Equation (17). Formally,

$$\psi(\mathbf{w}) = \psi \cdot \mathbf{w}^\intercal = \psi \cdot \big(\mathbf{e}_{n_1+1} \mid \cdots \mid \mathbf{e}_{n_w+1}\big)^\intercal = \Big(\mathsf{I2B}_b(n_1) \| \ldots \| \mathsf{I2B}_b(n_t)\Big)^\intercal$$

Lemma 2. *For all $\mathbf{y} \in \mathbb{F}_2^r$, it holds $(\psi \circ \phi)(\mathbf{y}) = \mathbf{y}$,* i.e.*, ψ is the inverse of ϕ.*

Proof. Ad oculos, let $\mathbf{y} = (\mathbf{y}_1 \| \ldots \| \mathbf{y}_w)$.

$$\begin{aligned}(\psi \circ \phi)(\mathbf{y}) &= \psi\Big(\big(\mathbf{e}_{\mathsf{B2I}(\mathbf{y}_1)+1} \| \ldots \| \mathbf{e}_{\mathsf{B2I}(\mathbf{y}_w)+1}\big) \Big) \\ &= \Big(\big(\mathsf{I2B}_b \circ \mathsf{B2I}\big)(\mathbf{y}_1) \| \ldots \| \big(\mathsf{I2B}_b \circ \mathsf{B2I}\big)(\mathbf{y}_w) \Big) = (\mathbf{y}_1 \| \ldots \| \mathbf{y}_w) = \mathbf{y}\end{aligned}$$

□

Given the invertibility property, we are now able to further modify and fix the naïve approach presented in Eq. (16). For every $j \in \{1, \ldots, (t-1)\}$, let us rewrite the equation

by moving all the addends to the left-hand side. Formally,

$$\begin{aligned}\mathbf{A}_{x_i} \cdot \phi(\mathbf{y}^{i-1})^\intercal = \mathbf{y}^{i\intercal} &\iff \mathbf{A}_{x_i} \cdot \phi(\mathbf{y}^{i-1})^\intercal \oplus \mathbf{y}^{i\intercal} = \mathbf{0} \\ &\iff \mathbf{A}_{x_i} \cdot \phi(\mathbf{y}^{i-1})^\intercal \oplus \big(\psi \circ \phi\big)^\intercal(\mathbf{y}^i) = \mathbf{0} \\ &\iff \mathbf{A}_{x_i} \cdot \phi(\mathbf{y}^{i-1})^\intercal \oplus \psi \cdot \phi(\mathbf{y}^i)^\intercal = \mathbf{0} \end{aligned} \tag{18}$$

By rewriting Eq. (18) in Eq. (16), define $\widehat{\mathbf{M}} \in \mathbb{F}_2^{tr \times tn}$ and $\widehat{\mathbf{s}} \in \mathbb{F}_2^{tn}, \widehat{\mathbf{y}} \in \mathbb{F}_2^{tr}$ as:

$$\widehat{\mathbf{M}} \cdot \widehat{\mathbf{s}} := \begin{pmatrix} \mathbf{A}_{x_1} & \psi & & \\ & \ddots & \ddots & \\ & & \mathbf{A}_{x_{t-1}} & \psi \\ & & & \mathbf{A}_{x_t} \end{pmatrix} \cdot \begin{pmatrix} \phi(\mathbf{y}^0)^\intercal \\ \vdots \\ \phi(\mathbf{y}^{t-2})^\intercal \\ \phi(\mathbf{y}^{t-1})^\intercal \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{y}^{t\intercal} \end{pmatrix} =: \widehat{\mathbf{y}} \tag{19}$$

Proposition 7. *Let $\widehat{\mathbf{M}}, \widehat{\mathbf{s}}, \widehat{\mathbf{y}}$ as defined in Eq. (19). The related Stern language,*

$$\hat{L} = \left\{ \left(\widehat{\mathbf{M}}, \widehat{\mathbf{y}}\right) \;\middle|\; \exists\, \widehat{\mathbf{s}} : \; wt(\widehat{\mathbf{s}}) = wt \;\wedge\; \widehat{\mathbf{M}} \cdot \widehat{\mathbf{s}} = \widehat{\mathbf{y}} \right\}$$

is equivalent to the PRF evaluation language for PRF *of Eq. (15),* i.e.*,*

$$L_{\mathsf{PRF}} = \{(\mathbf{x}, \mathbf{y}) \mid \exists\, \mathbf{k} : \; \mathsf{PRF}(\mathbf{k}, \mathbf{x}) = \mathbf{y}\}$$

Proof. Since the global parameters n, r, w are known, the matrix ψ is defined and it is trivial to observe that $\widehat{\mathbf{M}}$ can be reconstructed with the knowledge of the matrices $\mathbf{A}_0, \mathbf{A}_1$ and $\mathbf{x}$. Furthermore, since $t-1$ components of $\widehat{\mathbf{y}}$ are zero, only $\mathbf{y}^t$ is needed to correctly reconstruct the language statement's vector. To this point, we can rewrite $\hat{L}$ as:

$$\hat{L} = \left\{ \left((\mathbf{A}_0, \mathbf{A}_1, \mathbf{x}), \mathbf{y}^t\right) \;\middle|\; \exists\, \widehat{\mathbf{s}} : \; \mathsf{wt}(\widehat{\mathbf{s}}) = wt \;\wedge\; \widehat{\mathbf{M}} \cdot \widehat{\mathbf{s}} = \widehat{\mathbf{y}} \right\}$$

Since the PRF is defined by the matrices $\mathbf{A}_0, \mathbf{A}_1$, $\mathbf{y}^t = \mathsf{PRF}(\mathbf{k}, \mathbf{x})$ and the matrix multiplication represents the GGM iterated PRF computations, we have

$$\hat{L} = L'_{\mathsf{PRF}} = \{(\mathbf{x}, \mathbf{y}) \mid \exists\, \widehat{\mathbf{s}} : \; \mathsf{wt}(\widehat{\mathbf{s}}) = wt \;\wedge\; \mathsf{PRF}(\mathbf{k}, \mathbf{x}) = \mathbf{y}\}$$

and we are left to prove that possessing the PRF secret key $\mathbf{k} \in \mathbb{F}_2^r$ is equivalent to knowing all the regular-words and partial evaluations $\mathbf{y}^j \in \mathbb{F}_2^n$ used in the GGM transformation, for all indexes $j \in \{0, \dots, (t-1)\}$.

Given that the maps ϕ and ψ are, together, a bijection between $\mathbb{F}_2^r$ and the regular word in $\mathbb{F}_2^n$ with weight w, it holds that it is irrelevant which representation is known. Trivially, the knowledge of $\mathbf{k} = \mathbf{y}^0$ allows the computation of all the other partial evaluations $\mathbf{y}^j$ for $j \in \{1, \dots, (t-1)\}$ and therefore it holds $L_{\mathsf{PRF}} \subseteq L'_{\mathsf{PRF}} = \hat{L}$. For the same reasons, it is possible to *"forget"* the partial evaluation and have $L_{\mathsf{PRF}} \supseteq L'_{\mathsf{PRF}}$. In conclusion, it holds that $\hat{L} = L_{\mathsf{PRF}}$. □

Corollary 2. *By Stern protocol's Theorem 4 and Proposition 7, executing Stern's protocol on $\left(\widehat{\mathbf{M}}, \widehat{\mathbf{s}}, \widehat{\mathbf{y}}\right)$ as defined in Eq. (19), produces a Zero-Knowledge PRF argument protocol based on the code-based PRF* PRF *of Section 3.*

5 Theoretical Analysis for Implementation Cost

In this section, we provide an application scenario in which our protocol could be employed and we discuss the protocol's communication costs.

Let us consider an employee and an employer that are willing to sign an agreement document that guarantees special treatment for the employee. Since they do not fully trust each other, they agree on a shared document. To bind the reached agreement, they ask a notary $\mathcal{N}$ to witness the signing phase, of both the employee and employer, and *publicly commit*, by signing, the content of the agreed-document. We assume that the signed and agreed document is made public. In this way, a notary is *fully liable* and, at any moment, anyone can take the signed document and let the notary testify on the agreement's trustworthiness. This scenario is quite common, whenever we consider *physical verification* of identities or signatures while, the first number-theoretic example is given by Adleman [Adl83] in 1983.

Let us now consider the case in which the notary $\mathcal{N}$ accepts to be liable in a *limited way*. More precisely, a verifier $\mathcal{V}$ can interact with $\mathcal{N}$ and ask to prove the agreed-document's correctness **but** $\mathcal{V}$ cannot use the interaction-transcript-of-the-protocol to further prove the document's correctness to other people.

This can be seen as the *whistle-blower's notary* problem and is depicted in Figure 29. Let us explain the scenario in detail, while employing our ZK protocol.

The clients prepare a document $\mathbf{x}$ containing all the info that they are willing to publish. The notary, in possess of a secret key $\mathbf{k}$, will verify the document's validity and he/she will publicly commit to the document with $\mathbf{y} = \mathsf{PRF}_{\mathbf{k}}(\mathbf{x})$. A verifier $\mathcal{V}$ will be able to verify the correctness of $(\mathbf{x}, \mathbf{y})$ by running the ZK PRF argument protocol with the notary $\mathcal{N}$. The zero-knowledge property imposes to the the notary that he **must** be collaborative **and guarantees** that $\mathcal{V}$ cannot use the proof-transcript and make $\mathcal{N}$ liable. This counter-intuitive second point is better understood when we change our point of view: $\mathcal{N}$ can *choose whom to prove to* and therefore he/she can interact with a trustworthy judge that is interested in the correctness of the document, while $\mathcal{N}$ can refuse to interact with strangers and avoid repercussions of any kind.

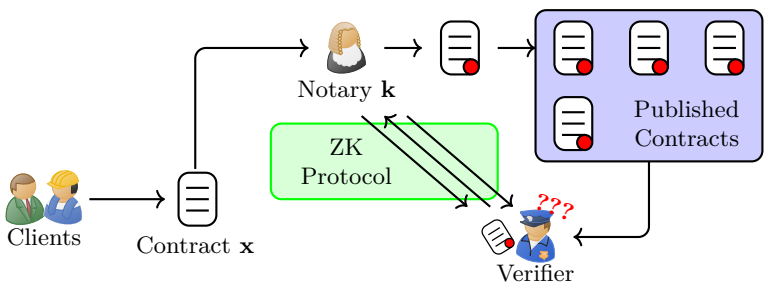


Figure 29: The whistle-blower notary problem.

In this way, *"committing"* and *"proving"* are done in different times. The reasons of this choice find roots in the extremely different cost between *"computing"* and *"communicating"* a statement or a proof. For this reason, we classify applications, such as the one described above, that are computationally-fast but communication-costly as ***prove-on-demand* protocols**, in which the protocol's communication cost is low **until** the proof is requested.

Let us now describe *why* our protocol is a prove-on-demand protocol. We upper-bound our ZK protocol communication cost *w.r.t.* implementation principles discussed

by Stern [Ste96] and Meziani *et al.* [MHC12].

First of all, we overestimate the length of a permutation π of the set $\{1, \dots, n\}$ as $|\pi| = n \log_2(n)$, which is the bit-representation of the permutation's image *w.r.t.* a fixed order, *e.g.*, $\pi = \big(\mathsf{I2B}_n(\pi(1)) \| \dots \| \mathsf{I2B}_n(\pi(n))\big)$. By employing the *random hashing technique*, as denoted by Stern, we may use a hash function H and commit to a message m by sampling some randomness ρ of the same length $|m|$ and commit by computing the hash value of $\big(\rho \| \rho \oplus m\big)$. To verify the decommitment, it is necessary to hold both ρ and m. In this way, the commitment's length is exactly the hash digest's length, denoted as $|\mathsf{H}|$.

Given $w, b \in \mathbb{N}$, the number d of Stern's protocol executions, and the PRF input space dimension t, the communication cost for our ZK PRF argument is:

$$\begin{aligned}\mathsf{Cost}(w, b, t, d) &= d \cdot \mathsf{Cost}_{\mathsf{Stern}}(tw2^b, twb) \\ &\leq d \cdot \Big(3\,|\mathsf{H}| + tw \cdot \Big(2^{b+1}\big(1 + b + \log_2(tw)\big) + b\Big) + 2\Big) \text{ bits}\end{aligned}$$

From the X-SYND definition [MHC12], in order to get a security level of 80 bits, the parameters are fixed as $w = 32$ and $b = 8$. We can also assume that the hash digest is $|\mathsf{H}| = 128$ bits. Therefore, if we consider t, d as parameters, we have:

$$\begin{aligned}\mathsf{Cost}(32, 8, t, d) &= d \cdot \mathsf{Cost}_{\mathsf{Stern}}(8192t, 256t) \\ &\leq d \cdot \Big(t \cdot \Big(16384 \cdot \log_2(t) + 229632\Big) + 386\Big) \text{ bits}\end{aligned}$$

With these parameters, we have that our proposed PRF has a space-cost equal to $|\mathbf{A}_0| + |\mathbf{A}_0| = 2 \cdot w2^b \cdot wb$ which, in our case, is 0.5 Megabyte. The output space is 256 bits. Although the matrices used in the computations require significant cost, our protocol and proposed primitives require only binary operations and thus have an extremely low communication cost.

It is clear that the communication cost is directly proportional to the soundness probability we want to achieve, *i.e.*, the probability of a successful adversary, who may want to impersonate the notary. For example, in order to get a soundness probability of less than 2^{-80}, we have to execute the protocol at least $d \geq 137$ times.
We plot the communication cost of running our ZK PRF argument protocol, depending on the (t, d) choices in Figure 30.

Regarding the PRF input space, we might consider, as a reasonable dimension, to be either $t = 128$ or $t = 256$, as the output space. In these two cases, the whole communication cost for proving the PRF evaluation would be in the order of approximately one Gigabyte.

$$\mathsf{Cost}(32, 8, 128, 137) \simeq 719.79\mathsf{MB} \qquad \mathsf{Cost}(32, 8, 256, 137) \simeq 1508.08\mathsf{MB}$$

Given the high-communication cost required, we would highly suggest the employment of our ZK PRF argument protocol only in *prove-on-demand* application scenarios *i.e.,* in applications where proving the PRF argument is not required frequently and thus, the communication cost, and related time, can be afforded without disrupting the application's functionality. We should note though that considering the great efficiency of the required computations, the protocol can be executed in devices with low computational abilities.

6 Conclusions and Future Work

In this paper, we construct the first zero-knowledge PRF argument based on the regular syndrome decoding assumption. Our construction starts from defining a PRG f, which

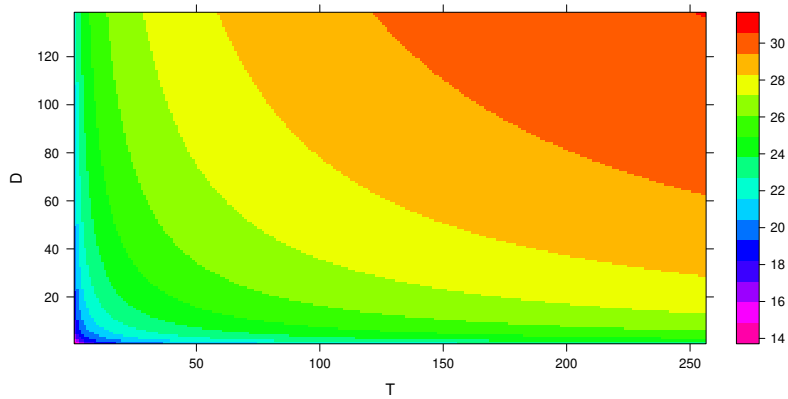


Figure 30: A heatmap plot of $\log_2\left(\frac{\mathsf{Cost}(32,8,t,d)}{8}\right)$ in which, for every t and d, we represent the communication cost in base-2 logarithmic scale. This means that a value of 20 represents 2^{20} bytes, which is 1 Megabyte.

directly reduces to a $\mathsf{RSD}(n, 2r, w)$ problem. By applying the GGM transformation we obtain a code-based PRF. After rewriting the GGM evaluation steps as a single linear system, we define the map ψ that consequently, allow us to rewrite the PRF evaluation in a Stern protocol statement. Finally, we obtain our code-based ZK PRF argument protocol by applying Stern's protocol.

Providing cryptographic primitives under code-based assumptions is of significant interest since code-based cryptography provides significant promise to be post-quantum secure. Furthermore, ZK PRF argument protocol can be employed to construct other code-based primitives. For instance, Brunetta *et al.*'s construction [BLM18] would allow us to define a *simulatable verifiable random function*, *i.e.*, a cryptographic primitive that allows to prove non-interactively the correct PRF computation. These advanced primitives can be used to simplify complex multi-party protocols employed in applications that require sampling a pseudorandom element from a set without allowing any party to maliciously affect the result, such as e-cash, e-voting and cryptographic lotteries.

As further work, we are interested in improving the proposed protocol's efficiency. Some possible directions, we may consider is improving Stern's protocol communication cost is by employing Aguilar *et al.*'s [AGS11] or Cayrel *et al.*'s [CVEYA11] protocols that also provide lower soundness error. Another direction is to reduce the PRF's *fingerprint* by using quasi-cycle codes and not random-binary ones.

Acknowledgement. We are grateful to the anonymous reviewers for their insightful comments. This work was partially supported by the Swedish Research Council (Vetenskapsrådet) through the grant PRECIS (621-2014-4845).

Towards Stronger Functional Signatures

Carlo Brunetta, Bei Liang and Aikaterini Mitrokotsa

Chalmers University of Technology, Gothenburg, Sweden

Manuscript

Abstract: *Functional digital Signatures* (FS) schemes introduced by Boyle, Goldwasser and Ivan (PKC 2014) provide a method to generate fine-grained digital signatures in which a master key-pair $(\mathsf{msk}, \mathsf{mvk})$ is used to generate a signing secret-key sk_f for a function f that allows to sign any message m into the message $f(\mathsf{m})$ and signature σ. The verification algorithm takes the master verification-key mvk and checks that the signature σ corresponding to $f(\mathsf{m})$ is valid. In this paper, we enhance the FS primitive by introducing a function public-key pk_f that acts as a commitment for the specific signing key sk_f. This public-key is used during the verification phase and guarantees that the message-signature pair is indeed the result generated by employing the specific key sk_f in the signature phase, a property not achieved by the original FS scheme. This enhanced FS scheme is defined as *Strong Functional Signatures* (SFS) for which we define the properties of unforgeability as well as the function hiding property. Finally, we provide an unforgeable, function hiding SFS instance in the random oracle model based on Boneh-Lynn-Shacham signature scheme (ASIACRYPT 2001) and Fiore-Gennaro's publicly verifiable computation scheme (CCS 2012).

Keywords: Functional Signatures, Verifiable Computation, Function Privacy

1 Introduction

Digital signatures, introduced by Diffie and Hellman [DH76], is a valuable cryptographic primitive that provides important integrity guarantees, *i.e.,* a signed message allows the receiver to verify that the message was indeed signed by the claimed signer. *Functional digital signatures* (FS), introduced by Boyle, Goldwasser and Ivan [BGI14] as a general extension of classic digital signatures [GMR88], allow generating signatures in a more *fine-grained manner*; thus, being very useful in multiple applications, *e.g.*, scenarios where the *delegation of signing rights* has to be considered. Functional digital signatures require a *trusted authority* to hold a *master secret key.* Given a description of a function f, the authority, using the master secret key, can generate a limited functional signing key sk_f associated with the function f. Anyone that has access to the signing key sk_f and a message m, can compute $f(\mathsf{m})$ **and** the corresponding *functional signature* σ of $f(\mathsf{m})$.

Let us employ an example related to photo-processing given by Boyle *et al.* [BGI14] to explain how FS works. When performing photo-processing, a digital camera is required to produce signed photos. One may want to allow photo-processing software to perform minor touch-ups of the photos, such as changing the contrast, but not allow more significant changes such as merging two photos or cropping a photo. Boyle *et al.* argued that FS could be used in such a setting to provide the photo processing software with a restricted key, which enables it to sign only specific modifications of an original photo. Let us assume there are three different pictures partitioned into three areas and coloured in red, blue and yellow but in different order, as represented in Figure 31.

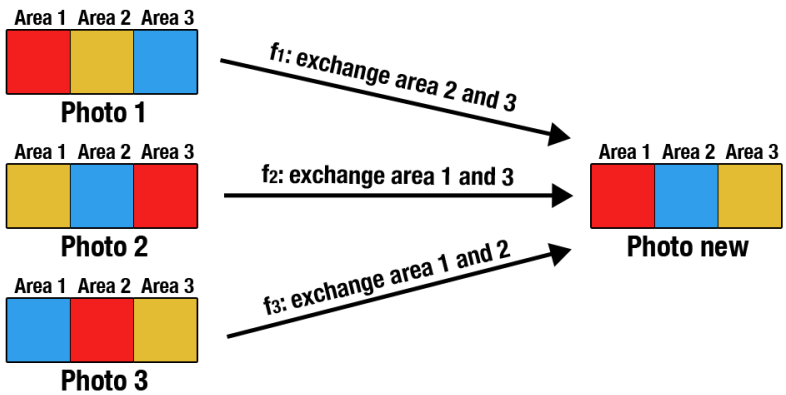


Figure 31: An illustrated example of collisions from different messages and functions in a *functional signature* scheme.

The functionality of f_1 is to exchange the colour of areas 2 and 3, while f_2 is used to exchange the colour of areas 1 and 3, and f_3 to exchange the colour of areas 1 and 2. Using the secret key sk_{f_1} to sign the photo ϕ_1, we obtain the signed new photo y_1. With the restricted keys sk_{f_2} and sk_{f_3}, we can obtain two signed photos with the same picture on it, namely y_2 and y_3. Using functional signatures, given y_1, y_2 and y_3, the appreciator (not the one who provides the original picture) only knows they are three certified photos.

Generally, if we consider two functions f and g and two messages m, m' such that $f(\mathsf{m}) = g(\mathsf{m}') = y$, then, given y and the corresponding functional signature σ, FS cannot be used to certify that the function value y is indeed computed from the queried

function f and m rather than from g and m'. The latter yields from the *function privacy* property of FS [BGI14], namely given y and σ, any adversary is unable to tell which function f or g was used to compute the value y even when given both functional signing keys sk_f and sk_g.

What if we wish to make the appreciator classify that a signed photo y, is indeed the outcome of applying an "allowed" *function without revealing* "which" *one?*

Our idea to allow an appreciator/verifier to distinguish between the usage of different secret keys, *e.g.* sk_f and sk_g, we introduce a *function public key*, *i.e.* pk_f and pk_g, that is just used in the verification phase. The public key pk_f can be seen as a commitment for the specific and related secret key sk_f allowing to distinguish between the evaluation and signatures $(f(\mathsf{m}), \sigma_1)$ and $(g(\mathsf{m}'), \sigma_2)$ even in the case that $f(\mathsf{m}) = g(\mathsf{m}')$. This *"key-addition"* directly affects the FS function privacy property that changes from *"the verifier cannot retrieve which function was computed"* to the stronger concept of *"the verifier cannot retrieve which function was computed* ***despite*** *knowing the related public key"*. We capture this idea into the enhanced definition of Strong Functional Signature (SFS), an Functional Signature (FS)-like scheme with function public keys that allows the verification of function evaluations' signatures **and** guarantees the correct function evaluation **while** maintaining the function hidden.

Example - Computational Authorisation for Cloud Computing our SFS primitive could be used in the example previously described, as well as in more general applications related to the cloud-assisted setting which are alike to the certification authorities' infrastructure **but** for function application and not only for identity authentication.

As depicted in Fig. 32, let us consider a cloud service $\mathcal{T}$ that offers to service providers $\mathcal{S}_i$ the possibilities to *register* their functionalities f_i in exchange of guaranteeing function hiding and the correct authentication whenever a user $\mathcal{U}_j$ wants to verify the authenticity and correctness of the output of such hidden functionalities. In other words, $\mathcal{S}_i$ will register the function f_i, obtain sk_{f_i} from $\mathcal{T}$ and, at the same time, $\mathcal{T}$ will publish the public key pk_{f_1} with some application label, *e.g.* it might be published into an *"Authorised"* functionality list. Later on, the user $\mathcal{U}_j$ requires $\mathcal{S}_i$ to process their data, obtains the output y with signature σ and wants to verify that y is indeed *correctly computed by an authorised* function. Therefore, $\mathcal{U}_j$ obtains the list of authorised public keys pk_{f_i} and verifies that (y, σ) is valid by finding a public key pk_f that pass the SFS validation algorithm. Additionally, $\mathcal{U}_j$ is unable to infer the precise function f from the public key pk_f thus the cloud service $\mathcal{T}$ guarantees to the service provider $\mathcal{S}$ that the function is kept private.

Observe that the cloud service $\mathcal{T}$ has the power to modify the status of the public keys, *e.g.* a public key pk_g might be completely *"revoked"* by removing it from all the public key's lists.

It is obvious that FS [BGI14, BF14, BMS16] does not have the features of checking if the outcome is resulted from the authorised functions, neither achieves this concept of *"revocability"*. In fact, in FS, since only mvk is required to verify the validity of (y, σ), it is not possible to check if a specific function was applied to output y, while our SFS make it possible by providing restricted public keys *w.r.t.* each function, which are employed in the verification process.

Moreover, in traditional FS schemes, it is indeed impossible to *"revoke"* a specific signing key, since the verification process would always work. However, in our introduced SFS notion, by incorporating the public keys in the verification process, we are able to revoke the signing capability for a restricted signing key thus allowing the trusted third party that owns the master key pair, to create a more fine-grained control over the generated function key pairs.

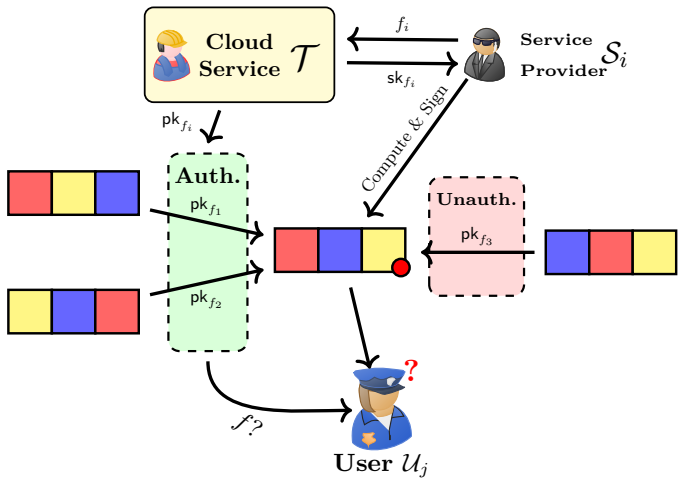


Figure 32: Strong functional signatures in the cloud computational authentication scenario.

Our Results our results can be summarised as follows:

- we formally define the notion of SFS with unforgeability and function hiding properties;
- we provide a variation of Boneh *et al.*'s BLS signature scheme [BLS04] and a variation of Fiore and Gennaro's verifiable computation scheme [FG12]. We prove that the Fiore and Gennaro's VC scheme satisfies the Public Verifiable Computation (PVC) privacy properties;
- based on our variations, we give an instantiation in the random oracle model of an SFS scheme for the polynomial function family which is adaptively unforgeable and satisfies the function hiding property.

The starting point of our instantiation of SFS is to use the BLS signature scheme [BLS04] in combination with the Fiore-Gennaro's publicly VC scheme [FG12] that is compatible with the algebraic structure and assumptions of the BLS signatures. We denote with $\overline{\mathsf{BLS}}$ the variation of the BLS signature and with $\overline{\mathsf{VC}}$ the variation of the Fiore-Gennaro's VC scheme, that we propose. The design-trick behind our instantiation is to *create* a *master key-pair* as an algebraic one-way instance and use it as a *"transposition"* for the secret key of the schemes, *e.g.* $\overline{\mathsf{BLS}}.\mathsf{Setup}(\lambda) \rightarrow (\mathsf{MSK}, \mathsf{MPK})$ is equal to $(\beta, e\,(g_1, g_2)^{\beta})$ for some $\beta \in \mathbb{Z}_p$ and whenever we sample a fresh secret value $\alpha \in \mathbb{Z}_p$ in order to compute the $\overline{\mathsf{BLS}}$ and the $\overline{\mathsf{VC}}$ keys, we just consider the new secret $\alpha + \beta$ obtained by translating α by β. Thus, all the evaluation/secret-keys are computed as if $\alpha+\beta$ is the randomness sampled while the verification/public-keys are published as *"local keys"*, *e.g.* we publish $e\,(g_1, g_2)^{\alpha}$ and not $e\,(g_1, g_2)^{\alpha+\beta}$. In this way, the two variated schemes become *"entangled"* thus implying a stricter relation during execution and verification. In a nutshell, the SFS instantiation combines the two schemes such that the verifiable computation $\overline{\mathsf{VC}}$ computes the secret function and provide the proof of correct computation while the signature scheme $\overline{\mathsf{BLS}}$ is used to sign the result and forcedly relate it to the $\overline{\mathsf{VC}}$ results.

Related Work SFS are inspired by Boyle *et al.* [BGI14] FS construction and are closely related to Signatures of Correct Computation (SCC) proposed by Papamanthou, Shi

and Tamassia [PST13] as well as PVC proposed by Parno *et al.* [PRV12] and Fiore and Gennaro [FG12].

Functional Signatures. This work is inspired by the notion of *Functional Signatures* (FS) introduced by Boyle *et al.* [BGI14]. They firstly proposed the formal definition of FS with *unforgeability security* as well as two additional desirable properties: *function privacy* and *succinctness*. Boyle *et al.* defined FS and gave a construction for an FS scheme, based on one-way functions and satisfying the unforgeability **but not** the succinctness or function privacy properties. Furthermore, they showed how to convert any FS without the function privacy or succinctness properties into an FS scheme that is succinct and function-private by using a SNARK scheme [GW11, BCCT12, BCCT13]. They also showed how to use an FS scheme to construct a delegation scheme [GGP10], *i.e.*, non-interactive verifiable computation.

Signatures of Correct Computation. Papamanthou, Shi and Tamassia introduced *Signatures of Correct Computation (SCC)* for verifying the correctness of a computation outsourced in the cloud [PST13]. In the SCC model, an authority wishes to outsource the execution of a function f to an untrusted server. It generates a pair of master keys along with a verification key $\mathsf{FK}(f)$ for that function which will be used during verification. Note that the existence of such a *verification key* for a function f and the requirement of being used for verification are similar to our formulation of SFS. The server can then return a signature σ on a value y, which certifies that the result y is indeed the correct outcome of the function f evaluated on some input. In the syntax of SCC [PST13], anyone with the public verification key can verify that an untrusted server correctly computed a function f on a specific input m. However, the verification algorithm requires the specific input m, used to compute $f(\mathsf{m})$, to be taken as input, which means that only the client or someone who knows the input m can verify the correctness of the computation. Therefore, SCC would not achieve any privacy with respect to the input m. In contrast, our SFS allows anyone to perform the verification without knowledge of the specific input m.

Publicly Verifiable Computation. Parno *et al.* [PRV12] have proposed a *publicly verifiable computation (PVC)* in which they consider a PVC scheme achieving two desirable properties: *public delegatability* and *public verifiability*. Their definition of PVC includes a $\mathsf{ProbGen}$ algorithm, which encodes a user's inputs m to a server's inputs σ_{m} and simultaneously prepares an element ρ_{m} to be used for verification. Thus, ρ_{m} can be used to publicly verify that the server returned a correct value. The *public delegation* property refers to the existence of a *public delegation key* pk_f *for the function* f, *i.e.*, the key used in the $\mathsf{ProbGen}$ algorithm, and publicly available to anyone. Thus, anyone can use the key and delegate the computation to the cloud.

Parno *et al.* [PRV12] also gave a construction of a VC scheme with public delegation and public verifiability from any *Attribute-Based Encryption (ABE)*, which is unfortunately not appropriate to be employed in order to instantiate a SFS since additional transformations are needed.

Another closely related work is the one by Fiore and Gennaro [FG12], who presented a very efficient PVC scheme tailored for multivariate polynomials over a finite field based on bilinear maps. We present a variation of their VC scheme by introducing a separate Setup algorithm to generate a master key pair for the scheme so that the keys for the evaluation of different functions could be executed multiple times using the same parameters for the scheme, which allows the evaluation of multiple functions on the same instance produced by $\mathsf{ProbGen}$.

Paper organisation. In Sec. 2, we describe the notations and review the primitives used in the paper. In Sec. 3, we propose two variances: one of Boneh *et al.*'s signature

scheme, denoted $\overline{\mathsf{BLS}}$, and one of the Fiore-Gennaro's PVC scheme, denoted $\overline{\mathsf{VC}}$. In Sec. 4, we provide the definition of SFS and its security properties and we instantiate an unforgeable and function hiding SFS using the $\overline{\mathsf{BLS}}$ and the $\overline{\mathsf{VC}}$ schemes.

2 Preliminaries

In the following section, we define the notations used through out the paper. We also provide the assumptions and the definitions of the building blocks that our constructions rely on.

2.1 Notations and Assumptions

In the paper, we denote with $x \leftarrow_R X$ the random uniform sampling in the set X, with λ the security parameter. We denote with $\mathbf{v}$ a vector and with $\mathbb{Z}_p$ the ring with p elements. When not specified, p always represents either a prime or a power of it. Let $\mathsf{Pr}[E]$ denote the probability that the event E occurs. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of the same order with generators g_1, g_2, g_T correspondingly and the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ of type-3, *i.e.* there does **not** exists an efficient homomorphism map $\psi : \mathbb{G}_2 \to \mathbb{G}_1$.

Definition 24 (co-Computation Diffie Hellman [BLS04, FG12])**.** *Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of prime order p. Let $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ be generators and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ bilinear map of type-3,* i.e. *there does not exists an efficient homomorphism map $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. We sample uniformly at random $a, b \leftarrow_R \mathbb{Z}_p$ and define the advantage of an adversary $\mathcal{A}$ in solving the **co-Computational Diffie Hellman** (co-CDH) problem as*

$$\mathsf{Adv}_{\mathcal{A}}^{co\text{-}CDH}(\lambda) = Pr\left[\mathcal{A}(p, g_1, g_2, g_1^a, g_2^b) = g_1^{ab}\right]$$

If for all adversaries $\mathcal{A}$ it exists a negligible ϵ such that $\mathsf{Adv}_{\mathcal{A}}^{co\text{-}CDH}(\lambda) \leq \epsilon$, then the co-CDH Assumption *ϵ-holds for the groups $\mathbb{G}_1, \mathbb{G}_2$.*

2.2 Closed Form Efficient PRFs

A closed form efficient PRF (Closed Form Efficient (CFE)-Pseudo Random Function (PRF)), defined by Fiore and Gennaro [FG12] consists of three algorithms $\mathsf{CF.KGen}$, $\mathsf{CF.H}$ and $\mathsf{CF.Eval}$. $\mathsf{CF.KGen}$ takes as input a security parameter λ and outputs a secret key K, from the key space $\mathcal{K}$, and some public parameters pp that specify the domain $\mathcal{X}$ and range $\mathcal{Y}$ of the function. For a fixed secret key K, $\mathsf{CF.H}_K$ takes as input a value $x \in \mathcal{X}$ and outputs a value $y \in \mathcal{Y}$. It satisfies the *pseudo-randomness* property: for every PPT adversary $\mathcal{A}$, $(K, \mathsf{pp}) \leftarrow \mathsf{CF.KGen}(\lambda)$ and any random function $\xi : \mathcal{X} \to \mathcal{Y}$:

$$\epsilon_{\mathsf{PRF}} = \left| \begin{array}{r} \mathsf{Pr}\left[\mathcal{A}^{\mathsf{CF.H}_K(\cdot)}(\lambda, \mathsf{pp}) = 1\right] - \\ - \mathsf{Pr}\left[\mathcal{A}^{\xi(\cdot)}(\lambda, \mathsf{pp}) = 1\right] \end{array} \right| \leq \mathsf{negl}(\lambda)$$

Additionally, the scheme is required to achieve *closed form efficiency*: consider a generic computation ϕ that has as input l random values $R_1, \ldots, R_l \in \mathcal{Y}$ and a vector of m arbitrary values $\mathbf{x} = (x_1, \ldots, x_m)$. Assume that the fastest computation time that takes to compute $\phi(R_1, \ldots, R_l, x_1, \ldots, x_m)$ is T. Let $\mathbf{z} = (z_1, \ldots, z_l)$ be a l-tuple of arbitrary values in the domain $\mathcal{X}$. The $\mathsf{CF.PRF}$ is said to achieve *closed form efficiency* for $(\phi, \mathbf{z})$ if the algorithm $\mathsf{CF.Eval}$ has running time $o(T)$ and it holds

$$\mathsf{CF.Eval}_{(\phi,\mathbf{z})}(K, \mathbf{x}) = \phi(\mathsf{CF.H}_K(z_1), \ldots, \mathsf{CF.H}_K(z_l), x_1, \ldots, x_m)$$

Fiore and Gennaro [FG12] give constructions of closed form efficient PRFs for multivariate polynomials and matrix multiplication, based on the decision linear assumption.

2.3 Functional Signatures

Boyle *et al.* [BGI14] introduced *functional digital signatures* (FS), a cryptographic primitive that can be employed to achieve *signing delegation*.

Definition 25 (Functional Signature [BGI14])**.** *A Functional Signature scheme for a message space $\mathcal{M}$ and function family $\mathcal{F} = \{f : \mathcal{D}_f \to \mathcal{M}\}$ consists of the PPT algorithms $\mathsf{FS} = (\mathsf{FS.Setup}, \mathsf{FS.KGen}, \mathsf{FS.Sign}, \mathsf{FS.Ver})$ defined as:*

- $\mathsf{FS.Setup}(\lambda) \to (\overline{\mathsf{msk}}, \overline{\mathsf{mvk}})$: *the setup algorithm takes as input the security parameter λ and outputs the master signing key $\overline{\mathsf{msk}}$ and the master verification key $\overline{\mathsf{mvk}}$.*
- $\mathsf{FS.KGen}(\overline{\mathsf{msk}}, f) \to \overline{\mathsf{sk}}_f$: *the key generation algorithm takes as input the master signing key and a function $f \in \mathcal{F}$ and outputs a signing key $\overline{\mathsf{sk}}_f$.*
- $\mathsf{FS.Sign}(f, \overline{\mathsf{sk}}_f, m) \to (f(m), \overline{\sigma})$: *the signing algorithm takes as input the signing key for a function f and an input $m \in \mathcal{D}_f$, and outputs $f(m)$ and a signature $\overline{\sigma}$ of $f(m)$.*
- $\mathsf{FS.Ver}(\overline{\mathsf{mvk}}, m', \overline{\sigma}) \to \{0, 1\}$: *the verification algorithm takes as input the master verification key $\overline{\mathsf{mvk}}$, a message m' and a signature $\overline{\sigma}$, and outputs 1 if the signature is valid.*

The definition requires the following conditions to hold:

Correctness a Functional Signature (FS) scheme is *correct* if for all functions $f \in \mathcal{F}$, messages $\mathsf{m} \in \mathcal{D}_f$, $(\overline{\mathsf{msk}}, \overline{\mathsf{mvk}})$ obtained from $\mathsf{FS.Setup}(\lambda)$, $\overline{\mathsf{sk}}_f$ obtained from $\mathsf{FS.KGen}(\overline{\mathsf{msk}}, f)$ and $(\mathsf{m}', \overline{\sigma})$ obtained from $\mathsf{FS.Sign}(f, \overline{\mathsf{sk}}_f, \mathsf{m})$, it holds that $\mathsf{FS.Ver}(\overline{\mathsf{mvk}}, \mathsf{m}', \sigma) = 1$.

Succinctness there exists a polynomial $s(\cdot)$ such that for every $\lambda \in \mathbb{N}$, function $f \in \mathcal{F}$, message $\mathsf{m} \in \mathcal{D}_f$, master keys $(\overline{\mathsf{msk}}, \overline{\mathsf{mvk}}) \leftarrow \mathsf{FS.Setup}(\lambda)$, function key $\overline{\mathsf{sk}}_f$ obtained from $\mathsf{FS.KGen}(\overline{\mathsf{msk}}, f)$, and $(f(\mathsf{m}), \overline{\sigma}) \leftarrow \mathsf{FS.Sign}(\overline{\mathsf{sk}}_f, \mathsf{m})$, it holds with probability 1 that $|\overline{\sigma}| \leq s(\lambda, |f(\mathsf{m})|)$.

Unforgeability FS is *unforgeable* if the probability of any PPT algorithm $\mathcal{A}$ in the FS unforgeability experiment $\mathsf{Exp}^{\mathsf{FS.UNF}}_{\mathsf{FS}}(\mathcal{A})$, depicted in Figure 33, to output 1 is negligible. Namely,

$$\mathsf{Adv}^{\mathsf{FS.UNF}}_{\mathcal{A},\mathsf{FS}}(\lambda) = \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{FS.UNF}}_{\mathsf{FS}}(\mathcal{A}) = 1\right] \leq \mathsf{negl}(\lambda)$$

Function privacy FS is *function private* if the advantage of any PPT algorithm $\mathcal{A}$ in the FS function privacy experiment $\mathsf{Exp}^{\mathsf{FS.FPriv}}_{\mathsf{FS}}(\mathcal{A})$, depicted in Figure 33 is negligible. Namely,

$$\mathsf{Adv}^{\mathsf{FS.FPriv}}_{\mathcal{A},\mathsf{FS}}(\lambda) = \left|\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{FS.FPriv}}_{\mathsf{FS}}(\mathcal{A}) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

2.4 The BLS Signature Scheme

In this section, we will report the Boneh *et al.*'s signature scheme [BLS04].

Let $(p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map in the security parameter λ. Let $\mathsf{H} : \{0,1\}^* \to \mathbb{G}_1$ be a full-domain hash function. The BLS signature scheme [BLS04] with the message space $\mathcal{M} = \{0,1\}^*$ comprises of the following three algorithms:

- $\mathsf{BLS.KGen}(\lambda) \to (\mathsf{PK}, \mathsf{SK})$: given a security parameter λ, sample a secret value $\mathsf{SK} \leftarrow_{\$} \mathbb{Z}_p$ and compute as the public key $\mathsf{PK} = g_2^{\mathsf{SK}}$.

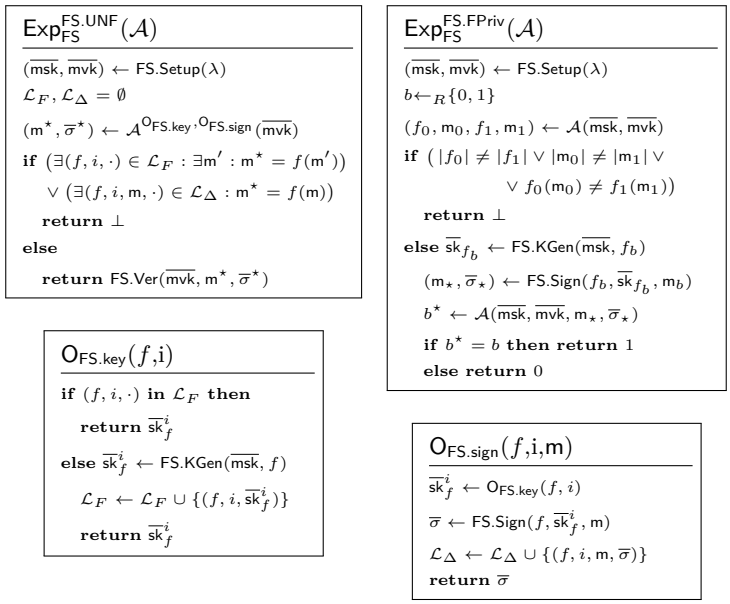


Figure 33: Functional signature unforgeability and function privacy experiments.

- $\mathsf{BLS.Sign}(\mathsf{SK}, \mathsf{m}) \rightarrow \ddot{\sigma}$: given a secret key SK and a message $\mathsf{m} \in \mathcal{M}$, compute $\mathsf{H}(\mathsf{m})$ and output the signature $\ddot{\sigma} = \mathsf{H}(\mathsf{m})^{\mathsf{SK}}$.

- $\mathsf{BLS.Ver}(\mathsf{PK}, \mathsf{m}, \ddot{\sigma}) \rightarrow \{0, 1\}$: given a public key PK, a message m and a signature $\ddot{\sigma}$, check $e(\ddot{\sigma}, g_2) \stackrel{?}{=} e(\mathsf{H}(\mathsf{m}), \mathsf{PK})$ and output 1 if it is true, otherwise output 0.

The BLS scheme is existentially unforgeable against chosen message attacks in the random oracle model (ROM), assuming the co-CDH assumption of Def. 24 holds.

2.5 Verifiable Computation

A *verifiable computation* (VC) scheme allows a client to delegate the computation of a function f to a server so that the client is able to verify the correctness of the result returned by the server with less computation cost than evaluating the function directly. We describe the definition of a *verifiable computation* (VC) scheme introduced by Parno *et al.* [PRV12] and Fiore and Gennaro [FG12].

Definition 26 (Verifiable Computation [PRV12, FG12])**.** *A verifiable computation scheme* VC *is defined by the following algorithms:*

- $\mathit{VC.KGen}(\lambda, f) \rightarrow (\widetilde{\mathsf{sk}}_f, \widetilde{\mathsf{vk}}_f, \widetilde{ek}_f)$ *: the key generation algorithm takes as input a security parameter* λ *and the description of a function* f*, and outputs a secret key* $\widetilde{\mathsf{sk}}_f$ *that will be used for input delegation, a corresponding verification key* $\widetilde{\mathsf{vk}}_f$*, and an evaluation key* $\widetilde{ek}_f$*, which will be used for the evaluation of* f*.*

- $\mathit{VC.ProbGen}(\widetilde{\mathsf{sk}}_f, m) \rightarrow (\widetilde{\sigma}_m, \widetilde{\rho}_m)$ *: the problem generation algorithm uses the secret key* $\widetilde{\mathsf{sk}}_f$ *to encode the function input* m *as an encoded value* $\widetilde{\sigma}_m$ *and a corresponding decoding value* $\widetilde{\rho}_m$*.*

- $\mathit{VC.Compute}(\widetilde{ek}_f, \widetilde{\sigma}_m) \to \widetilde{\sigma}_y$: *the computing algorithm takes as input the evaluation key* $\widetilde{ek}_f$ *and the encoded input* $\widetilde{\sigma}_m$ *and outputs* $\widetilde{\sigma}_y$*, an encoded version of the function's output* $y = f(m)$.
- $\mathit{VC.Ver}(\widetilde{vk}_f, \widetilde{\rho}_m, \widetilde{\sigma}_y) \to y$ or $\perp$: *the verification algorithm takes as input the verification key* $\widetilde{vk}_f$*, the decoding value* $\widetilde{\rho}_m$ *and the encoded output* $\widetilde{\sigma}_y$*. The algorithm outputs* y *if and only if* $y = f(m)$ *is correctly computed. Otherwise* $\perp$ *is the output.*

A publicly verifiable computation scheme is a VC scheme with an additional property that the verification key $\widetilde{vk}_f$ is published publicly such that anyone can check the correctness of a performed computation.

Remark 10. *The original VC [FG12] is with* "secret-key" *nature. In the earlier definition, KGen produces a secret key that was used as an input to ProbGen and, in turn, ProbGen produces a* secret *verification value needed for Ver. Later, Parno* et al. *[PRV12] introduced the* "public-key" *VC definition which has both the* public delegation ***and*** public verification *properties. The delegation being public or private depends on whether the evaluation key* $\widetilde{sk}$ *is published or kept secret. In our case, we consider the scenario where the Public Verifiable Computation (PVC) scheme is publicly verifiable but privately delegatable,* i.e. *the evaluation key* $\widetilde{ek}_f$ *is secret while the verification key* $\widetilde{vk}_f$ *is public. In the paper, we abuse terminology and refer to a PVC scheme when discussing about a Verifiable Computation (VC) scheme.*

Correctness a verifiable computation scheme VC is **correct** for a class of functions $\mathcal{F}$ if for any $f \in \mathcal{F}$, for any tuple of keys $(\widetilde{sk}_f, \widetilde{vk}_f, \widetilde{ek}_f) \leftarrow \mathsf{VC.KGen}(\lambda, f)$, for any $\mathsf{m} \in \mathcal{D}_f$, for any $(\widetilde{\sigma}_\mathsf{m}, \widetilde{\rho}_\mathsf{m}) \leftarrow \mathsf{VC.ProbGen}(\widetilde{sk}_f, \mathsf{m})$ and any computed $\widetilde{\sigma}_y$ obtained from $\mathsf{VC.Compute}(\widetilde{ek}_f, \widetilde{\sigma}_\mathsf{m})$, it holds that $\mathsf{VC.Ver}\big(\widetilde{vk}_f, \widetilde{\rho}_\mathsf{m}, \widetilde{\sigma}_y\big) = y = f(\mathsf{m})$.

Security a VC scheme is *secure w.r.t.* a static attacker if the probability of any PPT algorithm $\mathcal{A}$ in the VC static security experiment $\mathsf{Exp}^{\mathsf{VC.StaticVer}}_{\mathsf{VC}}(\mathcal{A})$ of Figure 34, to output 1 is negligible. Namely,

$$\mathsf{Adv}^{\mathsf{VC.StaticVer}}_{\mathcal{A},\mathsf{VC}}(\lambda) = \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{VC.StaticVer}}_{\mathsf{VC}}(\mathcal{A}) = 1\right] \leq \mathsf{negl}(\lambda)$$

Privacy [**FGP14**] a VC scheme is said to be *private w.r.t.* a static attacker if the advantage of any PPT algorithm $\mathcal{A}$ winning in the VC privacy experiment $\mathsf{Exp}^{\mathsf{VC.Priv}}_{\mathsf{VC}}(\mathcal{A})$ of Figure 34 is negligible. Namely,

$$\mathsf{Adv}^{\mathsf{VC.Priv}}_{\mathcal{A},\mathsf{VC}}(\lambda) = \left|\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{VC.Priv}}_{\mathsf{VC}}(\mathcal{A}) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

2.6 Fiore-Gennaro's PVC Scheme

Fiore and Gennaro [FG12] propose a publicly VC scheme for the function family $\mathcal{F}$ containing all multivariate polynomials $f(x_1, \ldots x_m)$ with coefficients in $\mathbb{Z}_p$ for some prime p, m variables and degree at most d in each variable. Let $h : \mathbb{Z}_p^m \to \mathbb{Z}_p^l$ which expands the input $\mathbf{x}$ to the vector $(h_1(\mathbf{x}), \ldots, h_l(\mathbf{x}))$ of all the monomials as follows: for all $j \in \{1, \ldots, l\}$ where $l = (d+1)^m$, write $j = (i_1, \ldots, i_m)$ with $i_k \in \{0, \ldots, d\}$, then $h_j(\mathbf{x}) = (x_1^{i_1} \cdots x_m^{i_m})$. Thus, by using this notation, it is possible to write the polynomial as $f(\mathbf{x}) = \langle \mathbf{f}, h(\mathbf{x}) \rangle = \sum_{j=1}^{l} f_j \cdot h_j(\mathbf{x})$ where the f_j's are its coefficients and $f_j \in \mathbb{Z}_p$. The construction works over the groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of the same prime order p, equipped with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let us define $\mathsf{Poly}(\mathbf{R}, \mathbf{x}) = \prod_{j=1}^{l} R_j^{h_j(\mathbf{x})}$ where $\mathbf{R}$ is a random l-dimensional vector of $\mathbb{G}_1^l$.

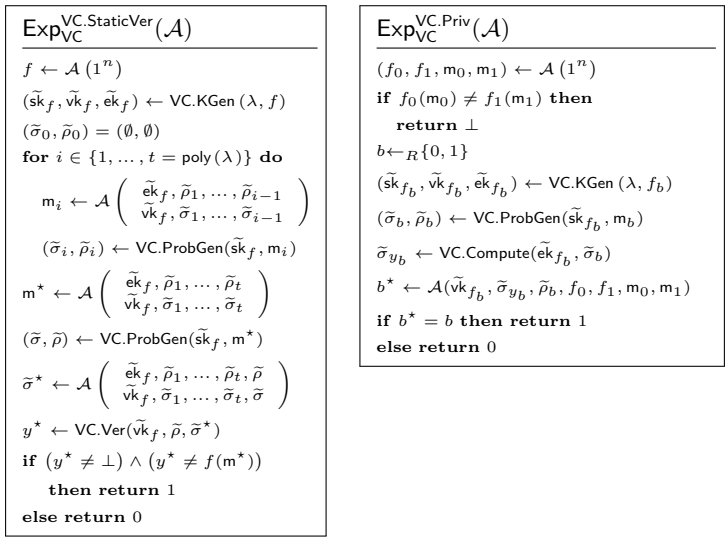


Figure 34: VC static security and privacy experiments.

Let $\mathsf{CF} = (\mathsf{CF.KGen}, \mathsf{CF.H}, \mathsf{CF.Eval})$ be a CFE PRF defined in Section 2.2. Fiore-Gennaro's public verifiable computation scheme [FG12] VC is constructed as the follows:

- $\mathsf{VC.KGen}(\lambda, f) \rightarrow (\widetilde{\mathsf{sk}}_f, \widetilde{\mathsf{vk}}_f, \widetilde{\mathsf{ek}}_f)$: Generate the description of a bilinear group $(p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ in the security parameter λ, a key of a PRF K with range in $\mathbb{G}_1$ as $K \leftarrow \mathsf{CF.KGen}(\lambda, \lceil \log d \rceil, m)$. Randomly sample $\alpha \leftarrow_R \mathbb{Z}_p$ and, for all the indexes $i \in \{1, \ldots, l\}$, compute $W_i = g_1^{\alpha \cdot f_i} \mathsf{CF.H}_K(i)$ and define W as $(W_1, \ldots, W_l) \in \mathbb{G}_1^l$. Output the key tuple $\left(\widetilde{\mathsf{sk}}_f, \widetilde{\mathsf{vk}}_f, \widetilde{\mathsf{ek}}_f\right)$ as the values $\left(K, e\,(g_1, g_2)^\alpha, (f, W)\right)$.

- $\mathsf{VC.ProbGen}(\widetilde{\mathsf{sk}}_f, \mathsf{m}) \rightarrow (\widetilde{\sigma}_\mathsf{m}, \widetilde{\rho}_\mathsf{m})$: Output the tuple $\left(\widetilde{\sigma}_\mathsf{m}, \widetilde{\rho}_\mathsf{m}\right)$ where $\widetilde{\sigma}_\mathsf{m} = \mathsf{m}$ and $\widetilde{\rho}_\mathsf{m} = e\,(\mathsf{CF.Eval}_{\mathsf{Poly}}(K, h(\mathsf{m})), g_2)$.

- $\mathsf{VC.Compute}(\widetilde{\mathsf{ek}}_f, \widetilde{\sigma}_\mathsf{m}) \rightarrow \widetilde{\sigma}_y$: Compute y by evaluating $f(\mathsf{m}) = \sum_{i=1}^{l} f_i h_i(\mathsf{m})$ and $V = \prod_{i=1}^{l} W_i^{h_i(\mathsf{m})}$. Output $\widetilde{\sigma}_y = (y, V)$.

- $\mathsf{VC.Ver}(\widetilde{\mathsf{vk}}_f, \widetilde{\rho}_\mathsf{m}, \widetilde{\sigma}_y) \rightarrow \{y, \perp\}$: output y if it holds that $e\,(V, g_2) \stackrel{?}{=} \left(\widetilde{\mathsf{vk}}_f\right)^y \cdot \widetilde{\rho}_\mathsf{m}$. Otherwise output $\perp$.

Fiore and Gennaro [FG12] proved that the construction is secure if the co-CDH assumption holds and $\mathsf{CF.PRF}$ is a close form efficient PRF. In Lemma 3, we prove that Fiore-Gennaro PVC scheme satisfies *privacy* as defined in the experiment depicted in Figure 34.

3 Construction Blocks: Variated Schemes

In this section, we provide our variations of the Boneh-Lynn-Shacham signature scheme [BLS04] and Fiore-Gennaro publicly verifiable computation scheme [FG12].

In a nutshell, the variations add to the schemes a *"setup algorithm"* that outputs a master key-pair used in the original key-generation algorithm and in the final verification algorithm while the accordingly modified security games reduce to the ones

of the original schemes. The final purpose of these modifications is to later allowing the instantiation of both the two schemes with a *single* common master key-pair in a stronger security setting, where the master secret-key is kept secure as in the act of *"merging"* the schemes into a single one. Intuitively, with the shared schemes' master public-key, the final verification algorithm will compute the two schemes' verification algorithms independently **and** will verify that the schemes are indeed *"merged"* into a single one.

3.1 A variation of the BLS signature

We introduce, in the BLS signature scheme, a Setup algorithm that outputs a master key-pair $(\mathsf{MPK}, \mathsf{MSK})$ used in the KeyGen algorithm to produce a local signing key in order to generate a signature for a message together with a local verification key. The Verify algorithm will take both the master public key and the local verification key to check the validity of a message-signature pair. We provide the unforgeability game for our BLS variation in Figure 35 and prove the unforgeability of it in the random oracle model.

Definition 27 (BLS Variation)**.** *Let $(p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map in the security parameter λ. Let $\mathsf{H} : \{0,1\}^* \to \mathbb{G}_1$ be a full-domain hash function and $F : \mathcal{K} \times \{0,1\}^* \to \mathbb{Z}_p$ a PRF. Let the additional information $\alpha \in \mathbb{Z}_p$ be a field element known just to the signer. Our variation $\overline{\mathsf{BLS}}$ scheme is defined as the algorithms:*

- $\overline{\mathsf{BLS}}.\mathsf{Setup}(\lambda) \to (\mathsf{MPK}, \mathsf{MSK})$*: sample $\beta \leftarrow\!\!\$\, \mathbb{Z}_p$ and set $\mathsf{MSK} = \beta$. Compute $\mathsf{MPK} = e(g_1, g_2)^{\beta}$ and output $(\mathsf{MPK}, \mathsf{MSK}) \in \mathbb{G}_T \times \mathbb{Z}_p$.*
- $\overline{\mathsf{BLS}}.\mathsf{KGen}(\mathsf{MSK}, \alpha) \to (\mathsf{PK}, \mathsf{SK})$*: given $\mathsf{MSK} \in \mathbb{Z}_p$ and $\alpha \in \mathbb{Z}_p$, sample $k \leftarrow\!\!\$\, \mathbb{Z}_p$, $r \in \mathbb{Z}_p$ and compute secret key as $\mathsf{SK} = (\mathsf{SK}_1, \mathsf{SK}_2) = (g_1^{\mathsf{MSK}+\alpha+r}, k)$ and the public key as $\mathsf{PK} = (\mathsf{PK}_1, \mathsf{PK}_2) = (e(g_1, g_2)^{\alpha+r}, g_2^{\mathsf{SK}_2})$.*
- $\overline{\mathsf{BLS}}.\mathsf{Sign}(\mathsf{SK}, m) \to \ddot{\sigma}$*: given a secret key $\mathsf{SK} = (\mathsf{SK}_1, \mathsf{SK}_2)$ and a message $m \in \mathcal{M}$, compute and output the signature $\ddot{\sigma} = \mathsf{SK}_1 \cdot H(m)^{\mathsf{SK}_2}$.*
- $\overline{\mathsf{BLS}}.\mathsf{Ver}(\mathsf{MPK}, \mathsf{PK}, m, \ddot{\sigma}) \to \{0,1\}$*: given a public key $\mathsf{PK} = (\mathsf{PK}_1, \mathsf{PK}_2)$, a message m, a signature $\ddot{\sigma}$ and a environmental public key MPK, verify and output the result of the check $e(\ddot{\sigma}, g_2) \stackrel{?}{=} \mathsf{MPK} \cdot \mathsf{PK}_1 \cdot e(H(m), \mathsf{PK}_2)$.*

$\mathsf{Exp}^{\mathsf{BLS.UNF}}_{\mathsf{BLS}}(\mathcal{A})$

$(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{BLS.KGen}(\lambda)$

$(\mathsf{m}^\star, \ddot{\sigma}^\star) \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{BLS.Sign}}(\mathsf{SK},\cdot),\mathsf{O}_{\mathsf{H}}(\cdot)}(\mathsf{PK})$

if $(\mathsf{m}^\star, \ddot{\sigma}^\star) \in \mathcal{L}_\Delta$ **then return** $\perp$

return $\mathsf{BLS.Ver}(\mathsf{PK}, \mathsf{m}^\star, \ddot{\sigma}^\star)$

$\mathsf{Exp}^{\overline{\mathsf{BLS}}.\mathsf{UNF}}_{\overline{\mathsf{BLS}}}(\mathcal{A})$

$(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \overline{\mathsf{BLS}}.\mathsf{Setup}(\lambda)$

$\alpha^\star \leftarrow \mathcal{A}(\mathsf{MPK})$; $\quad \mathcal{L}_\Delta = \emptyset$

$(\mathsf{PK}, \mathsf{SK}) \leftarrow \overline{\mathsf{BLS}}.\mathsf{KGen}(\lambda, \mathsf{MSK}, \alpha^\star)$

$(\mathsf{m}^\star, \ddot{\sigma}^\star) \leftarrow \mathcal{A}^{\mathsf{O}_{\overline{\mathsf{BLS}}.\mathsf{Sign}}(\mathsf{SK},\cdot),\mathsf{O}_{\mathsf{H}}(\cdot)}(\mathsf{MPK}, \mathsf{PK})$

if $(\mathsf{m}^\star, \ddot{\sigma}^\star) \in \mathcal{L}_\Delta$ **then return** $\perp$

return $\overline{\mathsf{BLS}}.\mathsf{Ver}(\mathsf{MPK}, \mathsf{PK}, \mathsf{m}^\star, \ddot{\sigma}^\star)$

Figure 35: BLS and $\overline{\mathsf{BLS}}$ unforgeability experiments.

We present in Fig. 35 a modified unforgeability experiment for the $\overline{\mathsf{BLS}}$ scheme which, differently from the BLS standard unforgeability experiment, must consider the

generation of the master key pair and the value $\alpha^\star$. We prove that, despite the modification, unforgeability is preserved.

Proposition 8. *If the advantage for all PPT adversaries $\mathcal{B}$ for the unforgeability experiment $\mathsf{Exp}_{\mathsf{BLS}}^{\mathsf{BLS.UNF}}(\mathcal{B})$ is negligible, then all the PPT adversaries $\mathcal{A}$ for the experiment $\mathsf{Exp}_{\overline{\mathsf{BLS}}}^{\overline{\mathsf{BLS}}.\mathsf{UNF}}(\mathcal{A})$ have a negligible advantage. Formally:*

$$\mathsf{Adv}_{\mathcal{A},\overline{\mathsf{BLS}}}^{\overline{\mathsf{BLS}}.\mathsf{UNF}}(\lambda) \leq \mathsf{Adv}_{\mathcal{B},\mathsf{BLS}}^{\mathsf{BLS.UNF}}(\lambda) \leq \mathsf{negl}(\lambda)$$

Proof. assume that there exists a PPT adversary $\mathcal{A}$ for the experiment $\mathsf{Exp}_{\overline{\mathsf{BLS}}}^{\overline{\mathsf{BLS}}.\mathsf{UNF}}(\mathcal{A})$ with non-negligible advantage Δ. The oracles $\mathsf{O}_{\mathsf{BLS.Sign(SK)}}(\mathsf{m})$ and $\mathsf{O}_{\overline{\mathsf{BLS}}.\mathsf{Sign(SK)}}(\mathsf{m})$ is to respond with the signatures on the messages m submitted to each challenger and then keep a track of the message-signature pair in its queried set $\mathcal{L}_\Delta$. Now we construct an adversary $\mathcal{R}$, running $\mathcal{A}$ as a subroutine, which attacks the underlying BLS scheme. Receiving from BLS challenger the public key $\mathsf{PK}_\star$, $\mathcal{R}$ sets it to be PK_2. $\mathcal{R}$ runs $\overline{\mathsf{BLS}}.\mathsf{Setup}(\lambda) \rightarrow (\mathsf{MPK}, \mathsf{MSK})$. It then outputs MPK to $\mathcal{A}$. $\mathcal{A}$ will reply with ξ and α. $\mathcal{R}$ fixes $\mathsf{SK}_1 = g_1^{\mathsf{MSK}+\alpha+r}$ and computes $\mathsf{PK}_1 = e\left(g_1^{\mathsf{MSK}+\alpha+r}, g_2\right)$ and outputs $\mathsf{PK} = (\mathsf{PK}_1, \mathsf{PK}_2)$ to $\mathcal{A}$. After the key generation phase, for every signing query $\mathsf{O}_{\overline{\mathsf{BLS}}.\mathsf{Sign}}(\mathsf{m})$ from $\mathcal{A}$, the reduction $\mathcal{R}$ queries $\mathcal{B}$'s oracle with $\mathsf{O}_{\mathsf{BLS.Sign}}(\mathsf{m})$ and obtains $\ddot{\sigma}_\star$. For any hash query $\mathsf{O}_{\mathsf{H}}(\mathsf{m})$ from $\mathcal{A}$, $\mathcal{R}$ queries $\mathcal{B}$'s hash oracle with $\mathsf{O}_{\mathsf{H}}(\mathsf{m})$ and obtains $\mathsf{H}(\mathsf{m})$. $\mathcal{R}$ computes $\ddot{\sigma} = \mathsf{SK}_1 \cdot \ddot{\sigma}_\star$ and returns it to $\mathcal{A}$. When $\mathcal{A}$ outputs the forgery $(\mathsf{m}^\star, \ddot{\sigma}^\star)$, the reduction $\mathcal{R}$ outputs $(\mathsf{m}^\star, \ddot{\sigma}^\star \cdot g_1^{-\alpha-\mathsf{MSK}-r})$.

It is direct to check that $\mathcal{R}$ output is a correct forgery for the BLS signature scheme since:

$$\begin{aligned}
&\mathsf{BLS.Ver}(\mathsf{PK}_\star, \mathsf{m}^\star, \ddot{\sigma}^\star \cdot g_1^{-\alpha-\mathsf{MSK}-r}) \Leftrightarrow \\
e\left(\ddot{\sigma}^\star \cdot g_1^{-\alpha-\mathsf{MSK}-r}, g_2\right) &\stackrel{?}{=} e\left(\mathsf{H}(\mathsf{m}^\star), \mathsf{PK}_\star\right) \Leftrightarrow \\
&\Leftrightarrow e\left(\ddot{\sigma}^\star, g_1\right) \stackrel{?}{=} e\left(g_1, g_2\right)^{\alpha+\mathsf{MSK}+r} \cdot e\left(\mathsf{H}(\mathsf{m}^\star), \mathsf{PK}_\star\right) \\
&\Leftrightarrow e\left(\ddot{\sigma}^\star, g_1\right) \stackrel{?}{=} \mathsf{MPK} \cdot \mathsf{PK}_1 \cdot e\left(\mathsf{H}(\mathsf{m}^\star), \mathsf{PK}_2\right) \\
&\Leftrightarrow \overline{\mathsf{BLS}}.\mathsf{Ver}(\mathsf{MPK}, \mathsf{PK}, \mathsf{m}^\star, \ddot{\sigma}^\star)
\end{aligned}$$

therefore $\Delta = \mathsf{Adv}_{\mathcal{A},\overline{\mathsf{BLS}}}^{\overline{\mathsf{BLS}}.\mathsf{UNF}}(\lambda) \leq \mathsf{Adv}_{\mathcal{B},\mathsf{BLS}}^{\mathsf{BLS.UNF}}(\lambda)$ which is a contradiction. □

3.2 A variation of Fiore-Gennaro's PVC

In our PVC variation, we introduce a *master key-pair* $(\widetilde{\mathsf{msk}}, \widetilde{\mathsf{mpk}})$ that is generated in the Setup phase and set as $(\beta, e\left(g_1, g_2\right)^\beta)$, which adds additional randomness to the evaluation key of function f such that W_i in Fiore-Gennaro's PVC is rerandomized to $W_i \cdot g_1^{\beta \cdot f_i}$. By forcing the master secret-key to be zero, *i.e.* $\beta = 0$, we obtain the original Fiore-Gennaro's scheme.

Definition 28 (Fiore-Gennaro PVC Variation)**.** *Let pp be the description of a bilinear group $(p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ in the security parameter λ. Our publicly verifiable computation scheme $\overline{\mathsf{VC}}$ is defined by the following algorithms:*

- $\overline{\mathsf{VC}}.\mathsf{Setup}(\lambda) \rightarrow (\widetilde{\mathsf{msk}}, \widetilde{\mathsf{mpk}})$: *the setup algorithm randomly sample $\beta \leftarrow_R \mathbb{Z}_p$ and outputs $\left(\widetilde{\mathsf{msk}}, \widetilde{\mathsf{mpk}}\right) = \left(\beta, e\left(g_1, g_2\right)^\beta\right)$.*

- $\overline{\mathsf{VC}}.\mathsf{KGen}(\lambda, \widetilde{\mathsf{msk}}, f) \rightarrow (\widetilde{\mathsf{sk}}_f, \widetilde{\mathsf{vk}}_f, \widetilde{\mathsf{ek}}_f)$: *let $\widetilde{\mathsf{msk}} = \beta$. The algorithm samples $\alpha \leftarrow_\$ \mathbb{Z}_p$ and generates a PRF key $K \leftarrow \mathsf{CF.KGen}(\lambda, \lceil \log d \rceil, m)$ with range in $\mathbb{G}_1$. For all $i \in \{1, \ldots, l\}$, it computes $W_i = g_1^{(\alpha+\beta) \cdot f_i} \mathsf{CF.H}_K(i)$ and let W be defined as $(W_1, \ldots, W_l) \in \mathbb{G}_1^l$. It outputs $\left(\widetilde{\mathsf{sk}}_f, \widetilde{\mathsf{vk}}_f, \widetilde{\mathsf{ek}}_f\right)$ as $\left((\alpha, g_2^\alpha, K), e\left(g_1, g_2\right)^\alpha, (f, W)\right)$.*

- $\overline{\mathsf{VC}}.\mathsf{ProbGen}(\widetilde{\mathsf{sk}}_f, m) \to (\widetilde{\sigma}_m, \widetilde{\rho}_m)$: *Output the tuple* $(\widetilde{\sigma}_m, \widetilde{\rho}_m)$ *where* $\widetilde{\sigma}_m = m$ *and* $\widetilde{\rho}_m = e\left(\mathsf{CF}.\mathsf{Eval}_{\mathsf{Poly}}(K, h(m)), g_2^{\alpha}\right)$.
- $\overline{\mathsf{VC}}.\mathsf{Compute}(\widetilde{\mathsf{ek}}_f, \widetilde{\sigma}_m) \to \widetilde{\sigma}_y$: *Compute* y *by evaluating* $f(m) = \sum_{i=1}^{l} f_i h_i(m)$ *and* $V = \prod_{i=1}^{l} W_i^{h_i(m)}$. *Output* $\widetilde{\sigma}_y = (y, V)$.
- $\overline{\mathsf{VC}}.\mathsf{Ver}(\widetilde{\mathsf{mpk}}, \widetilde{\mathsf{vk}}_f, \widetilde{\rho}_m, \widetilde{\sigma}_y) \to \{y, \perp\}$: *the algorithm checks if it holds that* $e(V, g_2) \stackrel{?}{=} \left(\widetilde{\mathsf{vk}}_f \cdot \widetilde{\mathsf{mpk}}\right)^y \cdot \widetilde{\rho}_m$. *If it is true, then it outputs* y. *Otherwise it outputs* $\perp$.

Remark 11. *It seems redundant to include* α *in* $\widetilde{\mathsf{sk}}_f$*, since the component of* (g_2^{α}, K) *suffices to obtain* $(\widetilde{\sigma}_m, \widetilde{\rho}_m)$*. However, looking ahead, the component* α *of* $\widetilde{\mathsf{sk}}_f$ *plays the role of building a bridge between* $\overline{\mathsf{VC}}$ *and* $\overline{\mathsf{BLS}}$ *in order to achieve an SFS.*

We describe the security and privacy experiments in Fig. 36.

$\mathsf{Exp}^{\overline{\mathsf{VC}}.\mathsf{StaticVer}}_{\overline{\mathsf{VC}}}(\mathcal{A})$

$(\boxed{\widetilde{\mathsf{mpk}}}, \widetilde{\mathsf{msk}}) \leftarrow \overline{\mathsf{VC}}.\mathsf{Setup}(\lambda)$

$f \leftarrow \mathcal{A}\left(\boxed{\widetilde{\mathsf{mpk}}}\right)$

$(\widetilde{\mathsf{sk}}_f, \widetilde{\mathsf{vk}}_f, \widetilde{\mathsf{ek}}_f) \leftarrow \overline{\mathsf{VC}}.\mathsf{KGen}\left(\begin{array}{c}\lambda, f \\ \boxed{\widetilde{\mathsf{msk}}}\end{array}\right)$

for $i \in \{1, \dots, t = \mathsf{poly}(\lambda)\}$ **do**

$\mathsf{m}_i \leftarrow \mathcal{A}\left(\begin{array}{c}\widetilde{\mathsf{ek}}_f, \widetilde{\rho}_1, \dots, \widetilde{\rho}_{i-1} \\ \widetilde{\mathsf{vk}}_f, \widetilde{\sigma}_1, \dots, \widetilde{\sigma}_{i-1}\end{array}, \boxed{\widetilde{\mathsf{mpk}}}\right)$

$(\widetilde{\sigma}_i, \widetilde{\rho}_i) \leftarrow \overline{\mathsf{VC}}.\mathsf{ProbGen}(\widetilde{\mathsf{sk}}_f, \mathsf{m}_i)$

$\mathsf{m}^{\star} \leftarrow \mathcal{A}\left(\begin{array}{c}\widetilde{\mathsf{ek}}_f, \widetilde{\rho}_1, \dots, \widetilde{\rho}_t \\ \widetilde{\mathsf{vk}}_f, \widetilde{\sigma}_1, \dots, \widetilde{\sigma}_t\end{array}, \boxed{\widetilde{\mathsf{mpk}}}\right)$

$(\widetilde{\sigma}, \widetilde{\rho}) \leftarrow \overline{\mathsf{VC}}.\mathsf{ProbGen}(\widetilde{\mathsf{sk}}_f, \mathsf{m}^{\star})$

$\widetilde{\sigma}^{\star} \leftarrow \mathcal{A}\left(\begin{array}{c}\widetilde{\mathsf{ek}}_f, \widetilde{\rho}_1, \dots, \widetilde{\rho}_t, \widetilde{\rho} \\ \widetilde{\mathsf{vk}}_f, \widetilde{\sigma}_1, \dots, \widetilde{\sigma}_t, \widetilde{\sigma}\end{array}, \boxed{\widetilde{\mathsf{mpk}}}\right)$

$y^{\star} \leftarrow \overline{\mathsf{VC}}.\mathsf{Ver}(\widetilde{\mathsf{vk}}_f, \widetilde{\rho}, \widetilde{\sigma}^{\star})$

if $(y^{\star} \neq \perp) \wedge (y^{\star} \neq f(\mathsf{m}^{\star}))$

then return 1

else return 0

$\mathsf{Exp}^{\overline{\mathsf{VC}}.\mathsf{Priv}}_{\overline{\mathsf{VC}}}(\mathcal{A})$

$(\boxed{\widetilde{\mathsf{mpk}}}, \widetilde{\mathsf{msk}}) \leftarrow \overline{\mathsf{VC}}.\mathsf{Setup}(\lambda)$

$b \leftarrow_R \{0, 1\}$

$(f_0, f_1, \mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}\left(\boxed{\widetilde{\mathsf{mpk}}}\right)$

if $f_0(\mathsf{m}_0) \neq f_1(\mathsf{m}_1)$ **then**

return $\perp$

$(\widetilde{\mathsf{sk}}_{f_b}, \widetilde{\mathsf{vk}}_{f_b}, \widetilde{\mathsf{ek}}_{f_b}) \leftarrow \mathsf{VC}.\mathsf{KGen}\left(\begin{array}{c}\lambda, f_b \\ \boxed{\widetilde{\mathsf{msk}}}\end{array}\right)$

$(\widetilde{\sigma}_b, \widetilde{\rho}_b) \leftarrow \mathsf{VC}.\mathsf{ProbGen}(\widetilde{\mathsf{sk}}_{f_b}, \mathsf{m}_b)$

$\widetilde{\sigma}_{y_b} \leftarrow \mathsf{VC}.\mathsf{Compute}(\widetilde{\mathsf{ek}}_{f_b}, \widetilde{\sigma}_b)$

$b^{\star} \leftarrow \mathcal{A}(\boxed{\widetilde{\mathsf{mpk}}}, \widetilde{\mathsf{vk}}_{f_b}, \widetilde{\sigma}_{y_b}, \widetilde{\rho}_b, f_0, f_1, \mathsf{m}_0, \mathsf{m}_1)$

if $b^{\star} = b$ **then return** 1

else return 0

Figure 36: The static security and privacy experiments for $\overline{\mathsf{VC}}$ scheme. In $\boxed{\text{box}}$ are highlighted the variations introduced in the $\overline{\mathsf{VC}}$ experiments in comparison to the original Fiore-Gennaro VC scheme.

Proposition 9. *If all PPT adversaries* $\mathcal{B}$ *for the experiment* $\mathsf{Exp}^{\mathsf{VC}.\mathsf{StaticVer}}_{\mathsf{VC}}(\mathcal{B})$ *have a negligible advantage, then all the PPT adversaries* $\mathcal{A}$ *for the experiment* $\mathsf{Exp}^{\overline{\mathsf{VC}}.\mathsf{StaticVer}}_{\overline{\mathsf{VC}}}(\mathcal{A})$ *have a negligible advantage. Formally:*

$$\mathsf{Adv}^{\overline{\mathsf{VC}}.\mathsf{StaticVer}}_{\mathcal{A},\overline{\mathsf{VC}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{VC}.\mathsf{StaticVer}}_{\mathcal{B},\mathsf{VC}}(\lambda) \leq \mathsf{negl}(\lambda)$$

and, mutatis mutandis*, it holds:*

$$\mathsf{Adv}^{\overline{\mathsf{VC}}.\mathsf{Priv}}_{\mathcal{A},\overline{\mathsf{VC}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{VC}.\mathsf{Priv}}_{\mathcal{B},\mathsf{VC}}(\lambda)$$

Proof. let us assume by contradiction that there exists a PPT adversary $\mathcal{A}$ for the experiment $\mathsf{Exp}^{\overline{\mathsf{VC}}.\mathsf{StaticVer}}_{\overline{\mathsf{VC}}}(\mathcal{A})$ with non-negligible advantage Δ. We build an adversary $\mathcal{R}$, running $\mathcal{A}$ as a subroutine, which attacks the security of the underlying VC scheme. $\mathcal{R}$ runs $\overline{\mathsf{VC}}.\mathsf{Setup}(\lambda) \rightarrow (\widetilde{\mathsf{mpk}}, \widetilde{\mathsf{msk}})$ and then outputs $\widetilde{\mathsf{mpk}}$ to $\mathcal{A}$ that will reply with the challenging function f. The reduction $\mathcal{R}$ just forwards it to the challenger of VC scheme and obtains $(\widetilde{\mathsf{vk}}_f, \widetilde{\mathsf{ek}}_f)$ where $\widetilde{\mathsf{ek}}_f = (f, W_\star)$. $\mathcal{R}$ modifies $W_\star$ into W by computing, for all $i \in \{1, \ldots, l\}$, the new values $W_i = W_{i\star} \cdot g_1^{\widetilde{\mathsf{msk}} \cdot f_i}$. It then returns $(\widetilde{\mathsf{vk}}_f, (f, W))$ to $\mathcal{A}$. All the $\mathsf{ProbGen}$ queries from $\mathcal{A}$ are just forwarded to the challenged of VC scheme and are responded with the same response from VC challenger. When the adversary $\mathcal{A}$ outputs the forgery $(i^\star, \widetilde{\sigma}^\star)$ where $\widetilde{\sigma}^\star = (y^\star, V^\star)$, the reduction $\mathcal{R}$ and outputs $(i^\star, (y^\star, V^\star \cdot g_1^{-\widetilde{\mathsf{msk}} \cdot y}))$. It is straightforward to check that $\mathcal{R}$ output is a correct tamper for the VC scheme since:

$$
\begin{aligned}
\mathsf{VC}.\mathsf{Ver}\left(\widetilde{\mathsf{vk}}_f, \widetilde{\rho}_{i^\star}, \left(y^\star, V^\star \cdot g_1^{-\widetilde{\mathsf{msk}} \cdot y}\right)\right) &\Leftrightarrow \\
e\left(V^\star \cdot g_1^{-\widetilde{\mathsf{msk}} \cdot y}, g_2\right) &\stackrel{?}{=} \widetilde{\mathsf{vk}}_f^{\,y} \cdot \widetilde{\rho}_{i^\star} \Leftrightarrow \\
e\left(V^\star, g_2\right) e\left(g_1, g_2\right)^{-\widetilde{\mathsf{msk}} \cdot y} &\stackrel{?}{=} \widetilde{\mathsf{vk}}_f^{\,y} \cdot \widetilde{\rho}_{i^\star} \Leftrightarrow \\
\Leftrightarrow e\left(V^\star, g_2\right) &\stackrel{?}{=} e\left(g_1, g_2\right)^{\widetilde{\mathsf{msk}} \cdot y} \cdot \widetilde{\mathsf{vk}}_f^{\,y} \cdot \widetilde{\rho}_{i^\star} \\
\Leftrightarrow e\left(V^\star, g_2\right) &\stackrel{?}{=} \left(\widetilde{\mathsf{mpk}} \cdot \widetilde{\mathsf{vk}}_f\right)^y \cdot \widetilde{\rho}_{i^\star} \\
\Leftrightarrow \overline{\mathsf{VC}}.\mathsf{Ver}(\widetilde{\mathsf{mpk}}, \widetilde{\mathsf{vk}}_f, \widetilde{\rho}_i^\star, \widetilde{\sigma}^\star)&
\end{aligned}
$$

therefore $\Delta = \mathsf{Adv}^{\overline{\mathsf{VC}}.\mathsf{StaticVer}}_{\mathcal{A}, \overline{\mathsf{VC}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{VC}.\mathsf{StaticVer}}_{\mathcal{R}, \mathsf{VC}}(\lambda) \leq \mathsf{negl}$ which is a contradiction. Similarly, it is easy to define a reduction $\mathcal{R}$ for an adversary $\mathcal{A}$ for the VC privacy experiments such that $\mathsf{Adv}^{\overline{\mathsf{VC}}.\mathsf{Priv}}_{\mathcal{A}, \overline{\mathsf{VC}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{VC}.\mathsf{Priv}}_{\mathcal{R}, \mathsf{VC}}(\lambda)$. □

We complement Fiore-Gennaro's results by providing the proof that their original VC scheme is indeed private, since this is needed to prove the function hiding property of the SFS construction.

Lemma 3. *If CF.*PRF *is a close form efficient PRF, then the Fiore-Gennaro PVC scheme is* private.

Proof. in order to prove the privacy of the Fiore-Gennaro scheme, we define a sequence of games that has the random bit b as input.

- Game_1(b, $\mathcal{A}$): the experiment $\mathsf{Exp}^{\mathsf{VC}.\mathsf{Priv}}_{\mathsf{VC}}(\mathcal{A})$ is executed by using the original Fiore-Gennaro scheme;
- Game_2(b, $\mathcal{A}$): in this game, the $\widetilde{\rho}_{\mathsf{m}_b}$ value is computed as

$$\widetilde{\rho}_{\mathsf{m}_b} = e\left(\prod_{i=1}^{l} \mathsf{CF}.\mathsf{H}_K(i)^{h_i(\mathsf{m_b})}, g_2\right);$$

- Game_3(b, $\mathcal{A}$): we exchange all the PRF evaluations $\mathsf{CF}.\mathsf{H}_K(i)$ with random elements R_i;
- Game_4(b, $\mathcal{A}$): we split the definition of W into a left and a right component $W = \{(W_{L_i}, W_{R_i})\}_{i=1}^{l} = \{(g_1^{\alpha f_{b\,i}}, R_i)\}_{i=1}^{l}$ and we substitute W_i with $W_{L_i} \cdot W_{R_i}$;
- Game_5(b, $\mathcal{A}$): after the challenge, we compute y which is equal to $f_0(\mathsf{m}_0) = f_1(\mathsf{m}_1)$, define $W_L = g_1^{\alpha \cdot y}$ and then substitute W with just the right component $W = \{W_{R_i}\}_{i=1}^{l}$. The game computes V as $W_L \cdot \prod_{i=1}^{l} R_i^{\,h_i(\mathsf{m}_b)}$

We highlight the difference between the games in Figure 37 in which we describe the challenger computations made after the challenger bit b sampling and before the bit b' guess. For compactness, we refer to $\mathsf{CF}.\mathsf{H}_K$ with just H_K and the notation $\{\cdot\}_i$ where the index i is contained in the set $\{1, \dots, l\}$.

$\mathsf{Game}_1(\mathrm{b})$

1 : $\left(K, e\left(g_1, g_2\right)^{\alpha}, \left(f_b, \left\{g_1^{\alpha \cdot f_{b_i}} \mathsf{H}_K(i)\right\}_i\right)\right)$

2 : $\left(\mathsf{m_b}, e\left(\mathsf{CF.Eval}_{\mathsf{Poly}}(K, h(\mathsf{m_b})), g_2\right)\right)$

3 : $\left(y, \prod_{i=1}^{l} W_i^{h_i(\mathsf{m}_b)}\right)$

$\mathsf{Game}_2(\mathrm{b})$

1 : $\left(K, e\left(g_1, g_2\right)^{\alpha}, \left(f_b, \left\{g_1^{\alpha \cdot f_{b_i}} \mathsf{H}_K(i)\right\}_i\right)\right)$

2 : $\left(\mathsf{m_b}, e\left(\prod_{i=1}^{l} \mathsf{H}_K(i)^{h_i(\mathsf{m_b})}, g_2\right)\right)$

3 : $\left(y, \prod_{i=1}^{l} W_i^{h_i(\mathsf{m}_b)}\right)$

$\mathsf{Game}_3(\mathrm{b})$

1 : $\left(e\left(g_1, g_2\right)^{\alpha}, \left(f_b, \left\{g_1^{\alpha \cdot f_{b_i}} R_i\right\}_i\right)\right)$

2 : $\left(\mathsf{m_b}, e\left(\prod_{i=1}^{l} {R_i}^{h_i(\mathsf{m_b})}, g_2\right)\right)$

3 : $\left(y, \prod_{i=1}^{l} W_i^{h_i(\mathsf{m}_b)}\right)$

$\mathsf{Game}_4(\mathrm{b})$

1 : $\left(e\left(g_1, g_2\right)^{\alpha}, \left(f_b, \left\{\left(g_1^{\alpha \cdot f_{b_i}} R_i\right)\right\}_i\right)\right)$

2 : $\left(\mathsf{m_b}, e\left(\prod_{i=1}^{l} {R_i}^{h_i(\mathsf{m_b})}, g_2\right)\right)$

3 : $\left(y, \prod_{i=1}^{l} \left(\left(g_1^{\alpha \cdot f_{b_i}}\right) \cdot R_i\right)^{h_i(\mathsf{m}_b)}\right)$

$\mathsf{Game}_5(\mathrm{b} = 1)$

1 : $y = f_1(\mathsf{m}_1)$

2 : $\left(e\left(g_1, g_2\right)^{\alpha}, \left(f_b, \left\{\left(g_1^{\alpha \cdot f_{b_i}} R_i\right)\right\}_i\right)\right)$

3 : $\left(\mathsf{m_1}, e\left(\prod_{i=1}^{l} {R_i}^{h_i(\mathsf{m_1})}, g_2\right)\right)$

4 : $\left(y, g_1^{\alpha \cdot y} \cdot \prod_{i=1}^{l} {R_i}^{h_i(\mathsf{m}_1)}\right)$

$\mathsf{Game}_5(\mathrm{b} = 0)$

1 : $y = f_0(\mathsf{m}_0)$

2 : $\left(e\left(g_1, g_2\right)^{\alpha}, \left(f_b, \left\{\left(g_1^{\alpha \cdot f_{b_i}} R_i'\right)\right\}_i\right)\right)$

3 : $\left(\mathsf{m_0}, e\left(\prod_{i=1}^{l} {R_i'}^{h_i(\mathsf{m_0})}, g_2\right)\right)$

4 : $\left(y, g_1^{\alpha \cdot y} \cdot \prod_{i=1}^{l} {R_i'}^{h_i(\mathsf{m}_0)}\right)$

Figure 37: The games used for proving the privacy of Fiore-Gennaro PVC scheme.

Claim 1. $Pr[\mathsf{Game}_1(b, \mathcal{A}) = 1] = Pr[\mathsf{Game}_2(b, \mathcal{A}) = 1]$

Proof. The only difference is on *"how to evaluate"* the $\mathsf{CF.Eval}_{\mathsf{Poly}}$ and by its correctness, the two are equivalent. □

Claim 2. $|Pr[\mathsf{Game}_2(b, \mathcal{A}) = 1] - Pr[\mathsf{Game}_3(b, \mathcal{A}) = 1]| \leq \epsilon_{\mathsf{PRF}}$

Proof. The difference between the games is that we replace the evaluation of the PRF with random elements. It is easy to see that an adversary $\mathcal{A}$ able to distinguish between the two games with non-negligible advantage can be used to define an adversary $\mathcal{B}$ able to distinguish the security of the $\mathsf{CF.PRF}$ with non-negligible advantage. □

Claim 3. $Pr[\mathsf{Game}_3(b, \mathcal{A}) = 1] = Pr[\mathsf{Game}_4(b, \mathcal{A}) = 1]$

Proof. The two games are equivalent since there is no difference between the two distributions. □

Claim 4. $Pr[\mathsf{Game}_4(b, \mathcal{A}) = 1] = Pr[\mathsf{Game}_5(b, \mathcal{A}) = 1]$

Proof. The difference between the two games is merely a computational optimisation since $\prod_{i=1}^{l} \left(g_1^{\alpha \cdot f_{b_i}}\right)^{h_i(\mathsf{m}_{b_i})} = g_1^{\alpha \cdot y}$ where $y = f_0(\mathsf{m}_0) = f_1(\mathsf{m}_1)$. Thus, there is no difference between the two games distributions. □

Claim 5. $Pr[\mathsf{Game}_5(1, \mathcal{A}) = 1] = Pr[\mathsf{Game}_5(0, \mathcal{A}) = 1]$

Proof. in order to prove the equality between the two probabilities, it is important to observe that, since the exponents $h_i(\mathsf{m_b})$ and $h_i(\mathsf{m_{1-b}})$ are fixed, the probability is measured on the random values R_i and R'_i. Fixed R_i, dually R'_i, there exists random values R'_i, dually R_i, such that the product $\prod_{i=1}^{l} {R_i}^{h_i(\mathsf{m}_b)}$ is equal to $\prod_{i=1}^{l} {R'_i}^{h_i(\mathsf{m}_{1-b})}$. Thus, by duality, the probabilities are the same. □

Therefore, the advantage is

$$
\begin{aligned}
&\mathsf{Adv}^{\mathsf{VC.StaticVer}}_{\mathcal{A},\mathsf{VC}}(\lambda) = \\
&= |\mathsf{Pr}\left[\mathsf{Game}_1(1, \mathcal{A}) = 1\right] - \mathsf{Pr}\left[\mathsf{Game}_1(0, \mathcal{A}) = 1\right]| \\
&\leq 2 \cdot \sum_{i=1}^{4} |\mathsf{Pr}\left[\mathsf{Game}_i(1, \mathcal{A}) = 1\right] - \mathsf{Pr}\left[\mathsf{Game}_{i+1}(1, \mathcal{A}) = 1\right]| + \\
&\quad + |\mathsf{Pr}\left[\mathsf{Game}_5(1, \mathcal{A}) = 1\right] - \mathsf{Pr}\left[\mathsf{Game}_5(0, \mathcal{A}) = 1\right]| \\
&\leq 2 \cdot \epsilon_{\mathsf{PRF}}
\end{aligned}
$$

□

4 Strong Functional Signatures

In this section, we define the Strong Functional Signature (SFS) primitive and the related unforgeability and function hiding experiments. We provide a specific SFS instantiation using the variated schemes introduced in Sec. 3 and prove it achieves unforgeability and function hiding.

4.1 SFS Definition

Our definition of an SFS scheme can be seen as a combination of a PVC and a FS scheme: similar to FS, an SFS scheme achieves delegation of the signing capability *w.r.t.* the master key-pair **and** it also allows the verification of the correct computation of the signing function f through an additional function public key pk_f, as a PVC scheme.

Definition 29 (Strong Functional Signature)**.** *A Strong Functional Signature (SFS) scheme for a message space $\mathcal{M}$ and function family $\mathcal{F}$ consists of the PPT algorithms* $\mathsf{SFS} = (\mathit{SFS.Setup}, \mathit{SFS.KGen}, \mathit{SFS.Sign}, \mathit{SFS.Ver})$ *defined as:*

- $\mathit{SFS.Setup}(\lambda) \rightarrow (\mathsf{msk}, \mathsf{mvk})$: *the setup algorithm takes as input the security parameter λ and outputs the master signing key and the master verification key.*
- $\mathit{SFS.KGen}(\mathsf{msk}, f) \rightarrow (\mathsf{pk}_f, \mathsf{sk}_f)$: *the key generation algorithm takes as input the master signing key and a function $f \in \mathcal{F}$ and outputs a secret signing key sk_f and a public verification key pk_f w.r.t. the function f.*
- $\mathit{SFS.Sign}(\mathsf{sk}_f, \mathsf{m}) \rightarrow (y, \sigma)$: *the signing algorithm takes as input the secret signing key for a function $f \in \mathcal{F}$ and a message in the function domain $\mathsf{m} \in \mathcal{D}_f$, and outputs a value $y = f(\mathsf{m})$ and a signature of $f(\mathsf{m})$.*

- $\mathit{SFS.Ver}(\mathsf{mvk}, \mathsf{pk}_f, y', \sigma) \to \{0,1\}$: *the verification algorithm takes as input the master verification key* mvk*, the public verification key* pk_f *for the function* f*, a message* y' *and a signature* σ*, and outputs* 1 *if the signature is valid and a correct computation of* f*,* 0 *if it is not a correct computation of* f *or the signature is not valid.*

We require the following conditions to hold:

Correctness for any function $f \in \mathcal{F}$, for any message $\mathsf{m} \in \mathcal{D}_f$, master keys $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{SFS.Setup}(\lambda)$, function keys $(\mathsf{pk}_f, \mathsf{sk}_f) \leftarrow \mathsf{SFS.KGen}(\mathsf{msk}, f)$, and (y, σ) obtained from $\mathsf{SFS.Sign}(\mathsf{sk}_f, \mathsf{m})$, it holds that $\mathsf{SFS.Ver}(\mathsf{mvk}, \mathsf{pk}_f, y, \sigma) = 1$.

Succinctness there exists a polynomial $s(\cdot)$ such that for every $\lambda \in \mathbb{N}$, function $f \in \mathcal{F}$, message $\mathsf{m} \in \mathcal{D}_f$, master keys $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{SFS.Setup}(\lambda)$, function keys $(\mathsf{pk}_f, \mathsf{sk}_f)$ obtained from $\mathsf{SFS.KGen}(\mathsf{msk}, f)$, and $(f(\mathsf{m}), \sigma) \leftarrow \mathsf{SFS.Sign}(\mathsf{sk}_f, \mathsf{m})$, it holds with probability 1 that $|\sigma| \leq s(\lambda, |f(\mathsf{m})|)$.

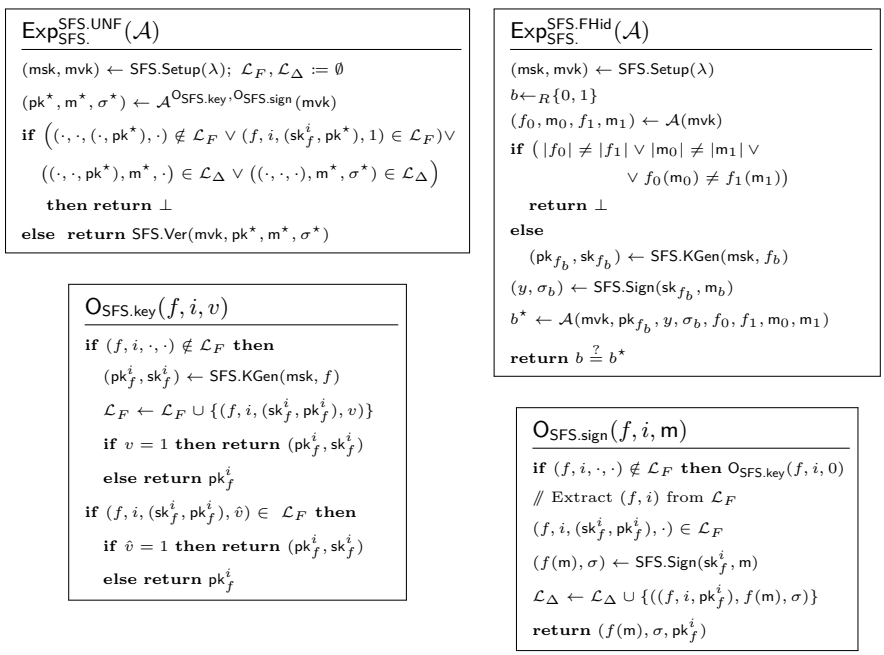


Figure 38: SFS unforgeability and function hiding experiments.

Unforgeability an SFS scheme is said to be *unforgeable* if the probability of any PPT algorithm $\mathcal{A}$ in the SFS unforgeability experiment $\mathsf{Exp}^{\mathsf{SFS.UNF}}_{\mathsf{SFS}}(\mathcal{A})$ depicted in Fig. 38 to output 1 is negligible. Namely,

$$\mathsf{Adv}^{\mathsf{SFS.UNF}}_{\mathcal{A},\mathsf{SFS}}(\lambda) = \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{SFS.UNF}}_{\mathsf{SFS}}(\mathcal{A}) = 1\right] \leq \mathsf{negl}(\lambda)$$

The main idea behind the unforgeability game is that an adversary $\mathcal{A}$ must present a tamper $(\mathsf{pk}^\star, \mathsf{m}^\star, \sigma^\star)$ for an existing honestly generated public key, whose corresponding secret key is not revealed to $\mathcal{A}$. We allow the adversary to arbitrarily request correct signatures and new key pairs that can be corrupted depending on the value of v, *i.e.* if $\mathcal{A}$

can obtain a corrupted key pair by querying $\mathsf{O}_{\mathsf{SFS.key}}(f, i, 1)$ where $v = 1$. We deliberately **do not allow** $\mathcal{A}$ to corrupt already generated key since this would imply that the third party that generates the function keys is able to identify whenever a specific public key is compromised. Despite being possible in the ideal world, this property is hard to realise in a realistic scenario thus we force $\mathcal{A}$ to declare at the generation, if a key pair is compromised or not.

Function Hiding an SFS scheme is said to be *function hiding* if the advantage of any PPT algorithm $\mathcal{A}$ in the SFS function hiding experiment $\mathsf{Exp}^{\mathsf{SFS.FHid}}_{\mathsf{SFS}}(\mathcal{A})$, of Figure 38 to output 1 is negligible. Namely,

$$\mathsf{Adv}^{\mathsf{SFS.FHid}}_{\mathcal{A},\mathsf{SFS}}(\lambda) = \left|\Pr\left[\mathsf{Exp}^{\mathsf{SFS.FHid}}_{\mathsf{SFS}}(\mathcal{A}) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

Informally, it is impossible for an adversary to distinguish between two different function evaluations and signatures, *i.e.*, given the public verification key of a single function, the adversary cannot infer information on *"what function does the key verify"*. When comparing to the FS function privacy property, the SFS function hiding requirement might appear *counter-intuitive* since, in the verification phase, it is necessary to use the **public**-key pk_f, which is related to the function f that must be hidden. The SFS function hiding property requires that *"a public-key should just allow the verification of the computation **but** must **not** provide any information of the function"*. This means that from a public-key pk_f, it must be hard to retrieve the corresponding function f.

4.2 An SFS Instantiation

In this subsection, we provide the instantiation of SFS scheme which is a combination of the Fiore-Gennaro's PVC variation (as given in Def. 28) and the BLS variation (as given in Def. 27).

Definition 30. *Let $\overline{\mathsf{BLS}}$ be the variated BLS signature scheme of Def. 27 and $\overline{\mathsf{VC}}$ the variated Fiore-Gennaro PVC scheme of Def. 28. Let the public parameter pp be the description of a bilinear group $(p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ shared between the $\overline{\mathsf{BLS}}$ and the $\overline{\mathsf{VC}}$ schemes. Define the SFS scheme for the polynomial function family $\mathcal{F}$, where every function can be expressed in a binary string representation, with the following algorithms:*

- $\mathsf{SFS.Setup}(\lambda) \to \mathsf{pp}, (\mathsf{msk}, \mathsf{mvk})$: *on input the security parameter λ, the algorithm runs $\overline{\mathsf{BLS}}.\mathsf{Setup}(\lambda) \to (\mathsf{MSK}, \mathsf{MPK})$, or equivalently $\overline{\mathsf{VC}}.\mathsf{Setup}$, and outputs the master key-pair $(\mathsf{msk}, \mathsf{mvk}) = (\mathsf{MSK}, \mathsf{MPK})$*
- $\mathsf{SFS.KGen}(\mathsf{msk}, f) \to (\mathsf{pk}_f, \mathsf{sk}_f)$: *on input the master secret key msk and a polynomial function f, execute $(\widetilde{\mathsf{sk}}_f, \widetilde{\mathsf{vk}}_f, \widetilde{\mathsf{ek}}_f) \leftarrow \overline{\mathsf{VC}}.\mathsf{KGen}(\mathsf{pp}, \mathsf{msk}, f)$, parse the secret key $\widetilde{\mathsf{sk}}_f = (\alpha, g_2^{\alpha}, K)$ and run the algorithm $(\mathsf{PK}_f, \mathsf{SK}_f) \leftarrow \overline{\mathsf{BLS}}.\mathsf{KGen}(\lambda, \mathsf{msk}, \alpha)$. Output $(\mathsf{pk}_f, \mathsf{sk}_f)$ defined as $\Big((\mathsf{PK}_f, \widetilde{\mathsf{vk}}_f), (\mathsf{SK}_f, (g_2^{\alpha}, K), \widetilde{\mathsf{ek}}_f)\Big)$*
- $\mathsf{SFS.Sign}(\mathsf{sk}_f, m) \to (y, \sigma)$: *given as input a secret key sk_f and a message m, parse $\mathsf{sk}_f = \big(\mathsf{SK}_f, (g_2^{\alpha}, K), \widetilde{\mathsf{ek}}_f\big)$ and execute $(\widetilde{\sigma}_m, \widetilde{\rho}_m) \leftarrow \overline{\mathsf{VC}}.\mathsf{ProbGen}((g_2^{\alpha}, K), m)$, then $\widetilde{\sigma}_y = (y, V) \leftarrow \overline{\mathsf{VC}}.\mathsf{Compute}(\widetilde{\mathsf{ek}}_f, \widetilde{\sigma}_m)$ and consequently compute the signature $\ddot{\sigma}_y \leftarrow \overline{\mathsf{BLS}}.\mathsf{Sign}(\mathsf{SK}_f, (y, \widetilde{\rho}_m, V))$. Output $(y, \sigma) = \big(y, (\widetilde{\rho}_m, V, \ddot{\sigma}_y)\big)$*
- $\mathsf{SFS.Ver}(\mathsf{mvk}, \mathsf{pk}_g, y', \sigma') \to \{0, 1\}$: *parse the inputs $\sigma' = (\widetilde{\rho}_{m'}, V, \ddot{\sigma}_{y'})$ and $\mathsf{pk}_g = (\mathsf{PK}_g, \widetilde{\mathsf{vk}}_g)$. Execute and output:*

$$\bigwedge \begin{array}{c} \overline{\mathsf{VC}}.\mathsf{Ver}(\mathsf{mvk}, \widetilde{\mathsf{vk}}_g, \widetilde{\rho}_{m'}, (y', V)) \stackrel{?}{=} y' \\ \overline{\mathsf{BLS}}.\mathsf{Ver}\big(\mathsf{mvk}, \mathsf{PK}_g, (y', \widetilde{\rho}'_m, V), \ddot{\sigma}_{y'}\big) \stackrel{?}{=} 1 \end{array}$$

Correctness for all $\mathsf{SFS.Setup}(\lambda) \to (\mathsf{msk}, \mathsf{mvk})$, functions $f \in \mathcal{F}$, $\mathsf{SFS.KGen}(\mathsf{msk}, f) \to (\mathsf{pk}_f, \mathsf{sk}_f)$ and messages m and $\mathsf{SFS.Sign}(\mathsf{sk}_f, \mathsf{m}) \to (y, \sigma)$, it holds $\mathsf{SFS.Ver}(\mathsf{mvk}, y, \sigma) = 1$ which translates into

$$\bigwedge \begin{array}{c} \overline{\mathsf{VC}}.\mathsf{Ver}(\mathsf{mvk}, \widetilde{\mathsf{vk}}_f, \widetilde{\rho}_\mathsf{m}, (y, V)) \stackrel{?}{=} y \\ \overline{\mathsf{BLS}}.\mathsf{Ver}\big(\mathsf{mvk}, \mathsf{PK}_f, (y, \widetilde{\rho}_\mathsf{m}, V), \ddot{\sigma}_y\big) \stackrel{?}{=} 1 \end{array}$$

and by correctness of the underlying $\overline{\mathsf{BLS}}$ and $\overline{\mathsf{VC}}$ scheme, it is indeed correct.

Succinctness we observe that the SFS's signature consists of three group elements and it is of constant size, *i.e.* $(\widetilde{\rho}_\mathsf{m}, V, \ddot{\sigma}_y) \in \mathbb{G}_T \times \mathbb{G}_1 \times \mathbb{G}_1$, thereby trivially achieving the succinctness property.

Unforgeability in order to prove our instantiation to be *unforgeable*, we will prove a reduction from the BLS unforgeability experiment $\mathsf{Exp}^{\mathsf{BLS.UNF}}_{\mathsf{BLS}}(\mathcal{B})$ to the SFS unforgeability experiment $\mathsf{Exp}^{\mathsf{SFS.UNF}}_{\mathsf{SFS}}(\mathcal{A})$.

Theorem 6. *If for all PPT adversaries $\mathcal{B}$ it holds that the advantage $\mathsf{Adv}^{\mathit{BLS.UNF}}_{\mathcal{B},\mathit{BLS}}(\lambda) \leq \mathsf{negl}(\lambda)$, then for all PPT adversaries $\mathcal{A}$ it holds $\mathsf{Adv}^{\mathit{SFS.UNF}}_{\mathcal{A},\mathit{SFS}}(\lambda) \leq \mathsf{negl}(\lambda)$.*

Proof. assume that there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{SFS.UNF}}_{\mathcal{A},\mathsf{SFS}}(\lambda) = \Delta$ for some non-negligible $\Delta > 0$. We construct an adversary $\mathcal{R}$, running $\mathcal{A}$ as a subroutine, to break the unforgeability of the underlying BLS scheme. $\mathcal{R}$ executes $\overline{\mathsf{VC}}.\mathsf{Setup}$ and obtains the master keys $(\mathsf{msk}, \mathsf{mvk})$. $\mathcal{R}$ receives from the BLS challenger the public key PK.

Whenever $\mathcal{A}$ queries a compromised key pair via $\mathsf{O}_{\mathsf{SFS.key}}(f, i, 1)$, $\mathcal{R}$ can generate the keys using $\overline{\mathsf{VC}}.\mathsf{KGen}$ and $\overline{\mathsf{BLS}}.\mathsf{KGen}$ and therefore can generate keys and compute the signing algorithm and answer to any adversarial signing query. On the other hand, whenever $\mathcal{A}$ queries a uncompromised pair $\mathsf{O}_{\mathsf{SFS.key}}(g, i, 0)$, $\mathcal{R}$ executes $\overline{\mathsf{VC}}.\mathsf{KGen}$ and generates the keys $(\widetilde{\mathsf{sk}}_g, \widetilde{\mathsf{ek}}_g, \widetilde{\mathsf{vk}}_g)$. $\mathcal{R}$ samples a random value $z_{(g,i)}$ sets the public key $\mathsf{PK}_2 = \mathsf{PK} \cdot g_2^{z_{(g,i)}}$.

By considering $\mathsf{MSK} = \mathsf{msk}$, $\mathcal{R}$ samples $\alpha, r \in \mathbb{Z}_p$, computes $\mathsf{SK}_1 = g_1^{\mathsf{MSK}+\alpha+r}$ and $\mathsf{PK}_1 = e\left(g_1^{\mathsf{MSK}+\alpha+r}, g_2\right)$ and obtains $\mathsf{PK}_g = (\mathsf{PK}_1, \mathsf{PK}_2)$. Finally, it sends $\mathsf{pk}_g = (\mathsf{PK}_g, \widetilde{\mathsf{vk}}_g)$ to $\mathcal{A}$.

In a nutshell, since the reduction $\mathcal{R}$ can create all the keys **except** the challenged SK, $\mathcal{R}$ is always able to correctly execute the verifiable computation scheme **but** not to sign the final output of a computation of any message m on the **uncompromised** functions g. This means that, whenever $\mathcal{A}$ queries the signing oracle $\mathsf{O}_{\mathsf{SFS.sign}}(g, i, \mathsf{m})$ for an uncompromised function (g, i), $\mathcal{R}$ will sequentially execute $\overline{\mathsf{VC}}.\mathsf{ProbGen}(\widetilde{\mathsf{sk}}_g, \mathsf{m})$ and the algorithm $\overline{\mathsf{VC}}.\mathsf{Compute}(\widetilde{\mathsf{sk}}_g, \widetilde{\sigma}_\mathsf{m})$ to obtain $\widetilde{\sigma}_y = (y, V)$ and $\widetilde{\rho}_\mathsf{m}$. At this point, $\mathcal{R}$ queries the BLS challenger on the message $(y, \widetilde{\rho}_\mathsf{m}, V)$ and obtains $\ddot{\sigma}$ which afterwards modifies into the value $\ddot{\sigma}_y = \mathsf{SK}_1 \cdot \ddot{\sigma} \cdot \mathsf{H}\big((y, \widetilde{\rho}_\mathsf{m}, V)\big)^{z_{(g,i)}}$. $\mathcal{R}$ replies to $\mathcal{A}$ with $(y, (\widetilde{\rho}_\mathsf{m}, V, \ddot{\sigma}_y))$.

Whenever $\mathcal{A}$ outputs the forgery $(\mathsf{pk}^\star, y^\star, \sigma^\star)$, the reduction $\mathcal{R}$ parses the output $\sigma^\star = (\widetilde{\rho}^\star, V^\star, \ddot{\sigma}^\star)$ and outputs the BLS forgery:

$$\Big((y^\star, \widetilde{\rho}^\star, V^\star),\ \ddot{\sigma}^\star \cdot \mathsf{SK}_1^{-1} \cdot \mathsf{H}\big((y^\star, \widetilde{\rho}^\star, V^\star)\big)^{-z_{(g,i)}}\Big)$$

Observe that $\mathcal{A}$ must output a forgery for an uncompromised function that, by construction, is always based on the challenged BLS scheme. The SFS unforgeability experiment's requirements forces $\mathcal{A}$ to always tamper at least one between $(y^\star, \sigma^\star)$ which always translates into $\mathcal{R}$ creating a new tamper never queried before to BLS. Thus, we can conclude that $\Delta = \mathsf{Adv}^{\mathsf{SFS.UNF}}_{\mathcal{A},\mathsf{SFS}}(\lambda) \leq \mathsf{Adv}^{\mathsf{BLS.UNF}}_{\mathcal{B},\mathsf{BLS}}(\lambda)$ which is a contradiction. □

Remark 12. *The unforgeability experiment* $\mathsf{Exp}^{\mathsf{SFS.UNF}}_{\mathsf{SFS.}}(\mathcal{A})$ *requires the adversary* $\mathcal{A}$ *to provide a tamper for a challenged public key* $\mathsf{pk}^\star$ *of a function* g *which* must exist *and be* uncompromised. *This means that* $\mathcal{A}$ *queried* $\mathcal{O}_{\mathsf{SFS.key}}(g, *, 0)$ *explicitly or implicitly via the signing oracle, and only owns the public key* $\mathsf{pk}^\star$.

As a matter of curiosity, Thm. 6's proof can be interpreted as the case where $\mathcal{A}$ *cannot forge even if the secret keys are* partially *compromised. In particular, consider that the proof's reduction* $\mathcal{R}$ *returns to* $\mathcal{A}$ ***all*** *the* $\overline{\mathsf{VC.KGen}}$ *generated keys* $(\widetilde{\mathsf{sk}}_g, \widetilde{\mathsf{ek}}_g, \widetilde{\mathsf{vk}}_g)$ *which would allow* $\mathcal{A}$ *to* always *pass the verification* $\overline{\mathsf{VC.Ver}}$. *Despite this additional concession, the proof shows that* $\mathcal{A}$ *is still unable to provide a tamper for* $\overline{\mathsf{BLS.}}$ *since* $\mathcal{A}$ *does not hold the* $\overline{\mathsf{BLS.}}$ *signing secret key, thus making it impossible to create a* SFS *tamper.*

Function Hiding in order to prove our instantiation to be *function hiding*, we will show a reduction from the VC function privacy experiment $\mathsf{Exp}^{\mathsf{VC.Priv}}_{\mathsf{VC}}(\mathcal{B})$ to the SFS function hiding experiment $\mathsf{Exp}^{\mathsf{SFS.FHid}}_{\mathsf{SFS}}(\mathcal{A})$.

Theorem 7. *If for all PPT adversaries* $\mathcal{B}$ *it holds that the advantage* $\mathsf{Adv}^{\mathsf{VC.Priv}}_{\mathcal{B},\mathsf{VC}}(\lambda) \leq \mathsf{negl}(\lambda)$*, then for all PPT adversaries* $\mathcal{A}$ *it holds* $\mathsf{Adv}^{\mathsf{SFS.FHid}}_{\mathcal{A},\mathsf{SFS}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Proof. assume the existence of a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{SFS.FHid}}_{\mathcal{A},\mathsf{SFS}}(\lambda) = \Delta$ for some non-negligible $\Delta > 0$. We then construct an adversary $\mathcal{B}$, running $\mathcal{A}$ as a subroutine, to break the privacy security of the underlying VC scheme. Let $\mathcal{R}$ be the reduction from the $\mathsf{VC.Priv}$ experiment to the $\mathsf{SFS.FHid}$ one and therefore $\mathcal{B}$ the final adversary that uses $\mathcal{R}$ and $\mathcal{A}$. $\mathcal{R}$ execute $\overline{\mathsf{VC.Setup}}(\lambda) \to (\widetilde{\mathsf{msk}}, \widetilde{\mathsf{mpk}})$ and sends $\mathsf{mvk} = \widetilde{\mathsf{mpk}}$ to the SFS adversary $\mathcal{A}$. $\mathcal{A}$ replies with the challenge $(f_0, \mathsf{m}_0, f_1, \mathsf{m}_1)$ which is forwarded to the $\mathsf{VC.Priv}$ challenger by $\mathcal{R}$. $\mathcal{R}$ receives $(\widetilde{\mathsf{vk}}_{f_b}, \widetilde{\sigma}_{y_b}, \widetilde{\rho}_b)$ where $\widetilde{\sigma}_{y_b} = (y, V_b)$ with y which is equal to $f_0(\mathsf{m}_0) = f_1(\mathsf{m}_1)$. $\mathcal{R}$ executes $\overline{\mathsf{BLS.KGen}}(\lambda, \mathsf{msk}, \alpha)$ for some random $\alpha \in \mathbb{G}$ and obtain $\mathsf{SK} = (\mathsf{SK}_1, \mathsf{SK}_2) = (g_1^{\mathsf{msk}+\alpha+r}, k)$ and $\mathsf{PK} = (\mathsf{PK}_1, \mathsf{PK}_2) = (e(g_1, g_2)^\alpha, g_2^k)$, then it signs $\overline{\mathsf{BLS.Sign}}(\mathsf{SK}, y)$ and obtains $\ddot{\sigma}$. The reduction $\mathcal{R}$ then replies to the $\mathcal{A}$ with the tuple $(\widetilde{\mathsf{vk}}_{f_b}, \widetilde{\sigma}^\star_{y_b}, \widetilde{\rho}_b, \mathsf{PK}, \ddot{\sigma})$ where $\widetilde{\sigma}^\star_{y_b} = (y, V_b{}^\star)$ which is equal to $(y, V_b \cdot g_1^{\mathsf{msk}\cdot y})$. Finally, $\mathcal{A}$'s guess is just forwarded to the challenger in VC's privacy game.

By observing the $\mathsf{SFS.Ver}$ algorithm, we get

$$\bigwedge \begin{array}{l} \overline{\mathsf{VC.Ver}}(\mathsf{mvk}, \widetilde{\mathsf{vk}}_{f_b}, \widetilde{\rho}_{\mathsf{m}_b}, (y, V_b{}^\star)) \stackrel{?}{=} y \\ \overline{\mathsf{BLS.Ver}}(\mathsf{mvk}, \mathsf{PK}, \overline{\mathsf{BLS.Sign}}(\mathsf{SK}, y)) \stackrel{?}{=} 1 \end{array}$$

and since the right side is always true, the left side is equivalent to

$$\begin{aligned} \overline{\mathsf{VC.Ver}}(\mathsf{mvk}, \widetilde{\mathsf{vk}}_{f_b}, \widetilde{\rho}_{\mathsf{m}_b}, (y, V_b{}^\star)) \quad &\iff \\ &\iff e(V_b{}^\star, g_2) \stackrel{?}{=} (\mathsf{mvk} \cdot \widetilde{\mathsf{vk}}_{f_b})^y \cdot \widetilde{\rho}_{\mathsf{m}_b} \\ &\iff e\left(V_b \cdot g_1^{\mathsf{msk}\cdot y}, g_2\right) \stackrel{?}{=} \mathsf{mvk}^y \cdot \widetilde{\mathsf{vk}}_{f_b}^{\,y} \cdot \widetilde{\rho}_{\mathsf{m}_b} \\ &\iff \mathsf{mvk}^y \cdot e(V_b, g_2) \stackrel{?}{=} \mathsf{mvk}^y \cdot \widetilde{\mathsf{vk}}_{f_b}^{\,y} \cdot \widetilde{\rho}_{\mathsf{m}_b} \\ &\iff e(V_b, g_2) \stackrel{?}{=} \widetilde{\mathsf{vk}}_{f_b}^{\,y} \cdot \widetilde{\rho}_{\mathsf{m}_b} \\ &\iff \mathsf{VC.Ver}(\widetilde{\mathsf{vk}}_{f_b}, \widetilde{\rho}_{\mathsf{m}_b}, (y, V_b)) \end{aligned}$$

Therefore, if the adversary $\mathcal{A}$ has an advantage Δ, the built adversary $\mathcal{B}$ for $\mathsf{VC.Priv}$ that uses $\mathcal{R}$ has advantage Δ. In other word, we conclude that $\Delta = \mathsf{Adv}^{\mathsf{SFS.FHid}}_{\mathcal{A},\mathsf{SFS}}(\lambda) \leq \mathsf{Adv}^{\mathsf{VC.Priv}}_{\mathcal{B},\mathsf{VC}}(\lambda)$ which is a contradiction. □

5 Conclusion

Verifying the correctness of computations is a very valuable property considering the ever-increasing cloud-assisted computing paradigm. This paper defines Strong Functional Signature (SFS) as an enhanced version of functional signatures with verifiable computation properties. In a nutshell, SFS introduce a functional public key pk_f that works as a commitment for a function f. This public-key allows in verification to guarantee the correct computation of the committed function without revealing any information on the function **and** to distinguish between different computed functions in a privacy-preserving way. Furthermore, we provide a concrete instantiation of an SFS scheme and prove that it satisfies the properties of *unforgeability* and *function hiding*.

5.1 Future Investigation (as of July 2021)

During the submission process, an anonymous reviewer brought to our attention a realistic attack not handled by the unforgeability experiment. The attacker $\mathcal{A}$ obtains a key pair $\Big(\big(\mathsf{PK}_f, \widetilde{\mathsf{vk}}_f\big), \big(\mathsf{SK}_f, (g_2^{\alpha}, K), \widetilde{\mathsf{ek}}_f\big)\Big)$ for a function f. Given the knowledge of K, $\mathcal{A}$ selects a different function g with decomposition $(g_1, \dots, g_l)$ and computes

$$W_i' = \left(\frac{W_i}{\mathsf{CF.H}_K(i)}\right)^{\frac{g_i}{f_i}} \cdot \mathsf{CF.H}_K(i) \qquad \forall\, i \in \{1, \dots, l\}$$

Observe that the W_i' are indeed the evaluation values that are used to evaluate the function g, *i.e.* $W_i' = g_1^{(\alpha+\beta)\cdot g_i}\mathsf{CF.H}_K(i)$. In this way, $\mathcal{A}$ creates a key pair for the evaluation of g and allows him/her to correctly sign $\mathsf{SFS.Sign}(\mathsf{sk}_g, \mathsf{m})$ and obtain a *correctly verifiable* output (y, σ) such that $g(\mathsf{m}) = y \neq f(\mathsf{m})$.

In other words, this attack allows an adversary that owns a secret key pair for f to sign *any function g evaluation*, making it impossible to correctly identify the computation correctness. Our unforgeability experiment of Fig. 38 does not incorporate such type/kind of attack as a valid forgery, because we exclude the case of forgery for any compromised secret key of function f. For this reason, we leave open for future development an augmented notion of unforgeability as intuitively represented in Fig. 39 and that incorporates the hypothesis of the FS unforgeability experiment $\mathsf{FS.UNF}$ of Fig. 33. The stronger unforgeability experiment would require the adversary to output a tamper $\mathsf{m}^\star$ that indeed is in the image of f. This implies that there exists a message m' such that $\mathsf{m}^\star = f(\mathsf{m}')$. As future work, we will consider how to give an instantiation that can achieve such an augmented security requirement.

$\mathsf{Exp}^{\mathsf{SFS.sUNF}}_{\mathsf{SFS.}}(\mathcal{A})$

$(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{SFS.Setup}(\lambda);\ \mathcal{L}_F, \mathcal{L}_\Delta := \emptyset$

$(\mathsf{pk}^\star, \mathsf{m}^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{SFS.key}}, \mathsf{O}_{\mathsf{SFS.sign}}}(\mathsf{mvk})$

if $\Big((\cdot,\cdot,(\cdot,\mathsf{pk}^\star),\cdot) \notin \mathcal{L}_F \vee (f, i, (\mathsf{sk}_f^i, \mathsf{pk}^\star), 1) \in \mathcal{L}_F) \vee$

$((\cdot,\cdot,\mathsf{pk}^\star), \mathsf{m}^\star, \cdot) \in \mathcal{L}_\Delta \vee ((\cdot,\cdot,\cdot), \mathsf{m}^\star, \sigma^\star) \in \mathcal{L}_\Delta\Big) \vee$

$\vee\ \boxed{(f, i, (\mathsf{sk}_f^i, \mathsf{pk}^\star), \cdot) \in \mathcal{L}_F, \exists \mathsf{m}' : \mathsf{m}^\star = f_i(\mathsf{m}')}$

then return $\perp$

else return $\mathsf{SFS.Ver}(\mathsf{mvk}, \mathsf{pk}^\star, \mathsf{m}^\star, \sigma^\star)$

Figure 39: SFS *stronger* unforgeability experiment.

Acknowledgment

We thank the anonymous reviewers for pointing out the security concern This work was partially supported by the Swedish Research Council (Vetenskapsrådet) through the grant PRECIS (621-2014-4845).

Modelling Cryptographic Distinguishers Using Machine Learning

Carlo Brunetta and Pablo Picazo-Sanchez

Chalmers University of Technology, Gothenburg, Sweden

Journal of Cryptographic Engineering
(July, 2021)

Abstract: Cryptanalysis is the development and study of attacks against cryptographic primitives and protocols. Many cryptographic properties rely on the difficulty of generating an adversary who, given an object sampled from one of two classes, correctly distinguishes the class used to generate that object. In the case of cipher suite distinguishing problem, the classes are two different cryptographic primitives. In this paper, we propose a methodology based on machine learning to automatically generate classifiers that can be used by an adversary to solve any distinguishing problem. We discuss the assumptions, a basic approach for improving the advantage of the adversary as well as a phenomenon that we call the *"blind spot paradox"*. We apply our methodology to generate distinguishers for the NIST Deterministic Random Bit Generators (DRBGs) cipher suite problem. Finally, we provide empirical evidence that the distinguishers might statistically have some advantage to distinguish between the DRBGs used.

Keywords: Cryptanalysis, Distinguisher, Machine Learning, Cipher Suite Distinguishing Problem, Pseudo Random Generator

1 Introduction

Nowadays, we use cryptography for almost all online activities, *e.g.* payments, secure messaging and web navigation. Even though cryptography is usually seen as *"one"* piece, this is not the case. Cryptographic protocols use different primitives to provide security and it is always required that each primitive needs to achieve a certain level of resistance against different attacks to be considered secure. Thus, it is crucial to define and classify *what an attack is*.

To define a cryptographic attack, we need to specify the *goal* and the *abilities* the adversary has *w.r.t.* a *security model* that describes how the primitive is used and attacked [KL08, Jou09]. For example, the plaintext recovering of an encrypted message without having the key is, by no means, the classical example of an attack on an encryption scheme.

Let us consider the concrete scenario in which an adversary is given an object sampled from one of two possible classes and the goal is to correctly guess the class used to generate that object. This adversary is known as a **distinguisher** for a *distinguishing problem*.

A classical example of a distinguishing problem in cryptography is the one used for the pseudorandomness property of a primitive (G) [Jou09]. Such a problem is defined as how to distinguish elements generated by G from those uniformly random generated, *i.e.* $(\mathsf{G}, \mathsf{rand})$. In other words, to prove the non-pseudorandomness, it would be sufficient to create a distinguisher D with a non-negligible advantage for solving the related distinguishing problem.

Machine Learning (ML) and cryptography have been widely combined in the literature, *e.g.* from random numbers generation [Koz91, ST96], random number prediction [KK06, KKG+09, PO14] and supervised algorithm using encrypted data [GAC18] to testing how good a Pseudo Random Generator (PRG) is [Fis18].

Our Contributions. In this paper, we propose a constructive methodology based on ML that allows the generation of several distinguishers $\{\mathsf{D}_i\}_i$ for a given distinguishing problem between two classes $(\mathsf{G}_0, \mathsf{G}_1)$. We implement a tool named MLCrypto and freely release the code to facilitate future work on this line[9].

In a nutshell, we generate a dataset that contains tuples of elements y_i together with the classes from which they are sampled. This dataset is the input of an ML algorithm whose outputs is a distinguisher D_i. It also generates strategies and solutions to allow an adversary to improve her advantage. Concretely, we present a strategy that allows us to combine several distinguishers $(\{\mathsf{D}_i\}_i)$ generated by MLCrypto to create a more accurate distinguisher D. We further discuss the *blind spot paradox*, a paradoxical phenomenon that can annihilate any advantage when the attacker unconscionably uses tools like MLCrypto in realistic attack scenarios.

We present a case study on the cipher suite distinguishing problem on the PRGs and based on the National Institute of Standard and Technology (NIST) DRBGs. We remark the state-of-the-art generation of a distinguisher from statistical test suites and link it with the advantage in breaking the pseudorandomness property. That is, having an advantage in discriminating between the PRG and a random element, with the advantage of distinguishing between two PRGs $(\mathsf{G}_0, \mathsf{G}_1)$. We design an experiment that uses MLCrypto as a distinguisher generator between DRBGs recommended by NIST [BK15].In more detail, MLCrypto generates Naive Bayes classifiers because of their *(i)* computational efficiency, *(ii)* implementation simplification, and; *(iii)* the lack of learning parameter to be tuned. From our experiments, we conclude that both our methodology and MLCrypto can be used for efficiently generating general purpose distinguishers.

[9] https://bitbucket.org/CharlieTrip/mlcryptocode/src/master/

Case study: distinguishing NIST DBRGs.

There are two main approaches to generate distinguishers: theoretical, and empirical. The theoretical approach consists of searching for flaws by scrutinizing the mathematical primitive definition. For instance, there are theoretical attacks [WS19, CKP+20] against PRGs proposed by NIST [BK15] based on specific differential cryptanalysis [BS91]. The empirical approach relies on defining a statistically significant number of experiments to provide enough confidence of the results, used to create a distinguisher. For instance, the test suite provided by NIST [BRS+10] is composed of multiple statistical tests that check whether the outputs generated by the PRG have some kind of correlation with the presence of some pattern—defined by each one of the tests. After running these tests, the outputs are compared to the result that a uniform distribution generates. The more passed tests, the more confidence in stating that the PRG is pseudorandom.

However, all these tests can, and more specifically the failed ones, be used to distinguish between PRGs and real randomness. By observing the failing tests, a distinguisher can infer that the input elements are generated by PRGs. They can be used to define fingerprints of the PRGs, *i.e.* each PRG is prone to fail the same tests, uniquely identifying them. Concretely, this distinguisher can be used to solve a related problem named *cipher suite distinguishing problem* [Jou09]. Similar to pseudorandomness, an attacker has to discriminate between objects generated by two different primitives $(\mathsf{G}_0, \mathsf{G}_1)$ and not from random elements.

Related work. Other works propose to distinguish between random numbers generated with block ciphers [DS06, HGDM+11, HZ19, ST13, ZZL18, Goh19, BBCD20] of which a vast majority extract features coming from the statistical tests proposed by NIST (NIST STS) [BRS+10] and use them as inputs of ML algorithms. While the documentation provided by the NIST does not provide any formal security analysis [Hir09], Woodgate *et al.* [WS19] carry out an in-depth security review. Contrarily to prior proposals, we apply MLCrypto to DRBGs recommended by NIST [BK15], being able to statistically distinguish between two pairs of generators.

To extract features from NIST STS to distinguish between random data generated from block ciphers, Zhao *et al.* [ZZL18] use Support Vector Machines (SVMs). They use OpenSSL to generate ciphertexts from AES, Camellia, Blowfish, DES, IDEA, and TDEA algorithms. Authors derive 54 features from the NIST STS, obtaining that accuracies of 42 features are higher than 50% while the accuracies of 12 features are higher than 60%. Hu *et al.* [HZ19] use random forest to classify random data from 16 block chipers instead of the 6 that Zhao *et al.* use, obtaining an accuracy of 88% in the classification. Svenda *et al.* [SUM13] use software circuits together with evolutionary algorithms to search for patterns, random bit predictability and random data indistinguishability.

Contrarily to the aforementioned works, instead of distinguishing between *random* data, we use MLCrypto as a machine learning approach to distinguish between *the functions* that generate these data, *i.e.* in our case study, we create distinguishers between NIST DRBGs [BK15].

Paper organisation. In Sec. 2, we give a brief introduction to pseudorandom generators, NIST DRBGs, and machine learning. Sec. 3 describes the methodology for generating distinguishers using machine learning and additionally discuss limitation, such as the blind spot paradox, and a possible strategy to amplify the adversarial advantage. In Sec. 4, we implement our methodology into the MLCrypto tool and consider a particular case-study based on DRBGs recommended by NIST. This paper ends with ideas for future work in Sec. 5.

2 Preliminaries

In this section, we present definitions and concepts used throughout the paper.

Notation. Let $\Pr_{x \in X}[E]$ denote the probability computed over the $x \in X$ that the event E occurs. We will omit the probability space whenever it is clear by the context, *i.e.* $\Pr[E]$. The random sampling in the set X is denoted as $x \leftarrow_R X$ and, whenever it is not specified, the sampling is always considered to be uniform at random. Let the natural number be denoted with $\mathbb{N}$, the real number field with $\mathbb{R}$ and the positive ones with $\mathbb{R}_+$. Let $[a, b]$ denote the interval between a and b, comprised. The space of binary strings of length ℓ is $\{0,1\}^\ell$ while $\|$ denotes binary concatenation.

Cryptography.

We report the definition of a Pseudo Random Generator (PRG) and the abstract NIST construction framework for a DRBG. For readability, we omit the error handling of these constructions.

Definition 31 (PRG [KL08])**.** *Given the positive integers $\ell_{in}, \ell_{out} \in \mathbb{N}$ with $\ell_{out} > \ell_{in}$, let $G : \{0,1\}^{\ell_{in}} \to \{0,1\}^{\ell_{out}}$ be a deterministic function. We say that G is a* ***pseudorandom generator*** *if the following two distributions are computationally indistinguishable for a distinguisher D:*

- *Sample a random seed $s \leftarrow_R \{0,1\}^{\ell_{in}}$ and output $G(s)$.*
- *Sample a random string $r \leftarrow_R \{0,1\}^{\ell_{out}}$ and output r.*

Definition 32 (Abstract NIST DRBG)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, $\widetilde{s} \in \{0,1\}^\lambda$ a bit string obtained by a random source, $\ell_s \in \mathbb{N}$ the seed length and $\ell_r \in \mathbb{N}$ the number of iterations before requiring the seed's reseed. We define a seed $\widetilde{s} \in \{0,1\}^{\ell_s}$, a nonce $\nu \in \{0,1\}^\lambda$ and an auxiliary string $\mathsf{aux} \in \{0,1\}^*$. Let a NIST abstract DRBG be defined by the algorithms:*

- $\mathsf{init}(\widetilde{s}, \nu, \mathsf{aux}, \lambda) \to \mathsf{state}_1$*: given a random binary string $\widetilde{s}$, a nonce ν, an auxiliary string aux and the security parameter λ, the instantiation algorithm outputs the initial internal stage state_1.*
- $\mathsf{reseed}(\mathsf{state}, \widetilde{s}', \mathsf{aux}) \to \mathsf{state}'_1$*: given an internal state state, a random binary string $\widetilde{s}$, an auxiliary binary string aux, the reseeding algorithm outputs a fresh initial internal stage state'_1.*
- $\mathsf{gen}(\mathsf{state}_i, n, \mathsf{aux}) \to (y, \mathsf{state}_{i+1})$*: given the internal state state_i, a non-zero number of output bit $n \in \mathbb{N}$ and an auxiliary string aux, the generation algorithm outputs the pseudorandom bit-string $y \in \{0,1\}^n$ and the successive internal stage state_{i+1}.*

The DRBG is defined as a state machine and it is depicted in Fig. 40. It takes a random binary string $\widetilde{s}$, a nonce ν, an auxiliary string aux and the security parameter λ to initialise the internal state and generates the internal state state_1. The internal state state_i is used as input of subsequent updates together with a non-zero number $n \in \mathbb{N}$ indicating the number of random bits requested and an auxiliary string aux. It outputs a n random bit-string y and updates the internal state to the next state state_{i+1}. Whenever it is requested, the DRBG can be reseeded, *i.e.* it starts again from a new internal state producing a new state'_1 given a previous state state_i, a new random binary string $\widetilde{s}'$ and some auxiliary information aux'.

To correctly instantiate the DRBG, the NIST suggests three different constructions: *1)* a *hash* function; *2)* the *HMAC* of a hash function, and; *3)* a *block cipher in counter-mode*. NIST requires the use of recommended cryptographic primitives [BK15], *e.g.*

HMAC with a secure hash function, AES-128 or SHA-2 family, **and** a bit string obtained by a secure random source [BK16, STBK+18]. Whenever it is not specified, we always consider NIST approved primitives and security parameters.

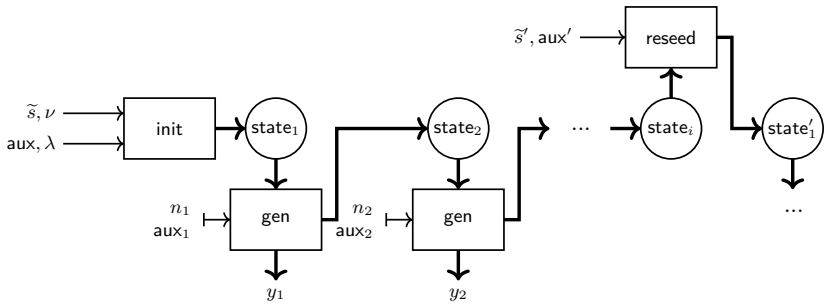


Figure 40: A state machine representation of the NIST DRBG work flow.

Machine Learning. Roughly speaking, ML is a set of algorithms whose goal is finding and describing patterns over a dataset. The dataset is usually composed of independent *instances* each one defined by a set of *features* or *attributes*. Once the dataset is generated, it is used as input of the ML that produces the knowledge that has been learnt [Alp14].

There are for main types of learning: *(i)* supervised learning or classification; *(ii)* unsupervised learning or clustering; *(iii)* association, and; *(iv)* numeric prediction [WF02]. In supervised learning, the ML learns from an already labelled dataset and it tries to predict the class of a new instance. On the contrary, in unsupervised learning, the dataset is not labelled and the ML algorithm looks for common patterns based on heuristics. Association seeks for relationships between the features of the dataset whereas the goal of the numeric prediction learning algorithms is to predict numbers instead of (labelled) data.

Naïve Bayes. The intuition behind Naïve Bayes is that features are independent and equally important. This is the consequence of applying the Bayes theorem into a classification algorithm. There is a particular case of Naïve Bayes algorithm when the likelihood of the features follows a Gaussian distribution, *i.e.* when the (continuous) values associated with each feature are distributed according to a Gaussian distribution.

3 Machine Learning Distinguishers

In this section, we formally define the distinguishing problem and present our methodology which explains how ML can be used to solve a distinguishing problem. We discuss how to use the accuracy we obtain from ML as a cryptographic advantage, a curious phenomenon we call *"blind spot paradox"* and propose a generic methodology to increase the advantage of a distinguisher at the cost of generating multiple ones.

In cryptography, it is common to find security properties defined by the probability of an adversary $\mathcal{A}$ being able to distinguish between **two** different instances. For example, in a simulation-based proof, $\mathcal{A}$ must discriminate between a real execution of a protocol and an ideal functionality assumed to be secure. Whenever proving the pseudorandomness of a function, $\mathcal{A}$ must choose if a value is computed by the function or if it is randomly sampled.

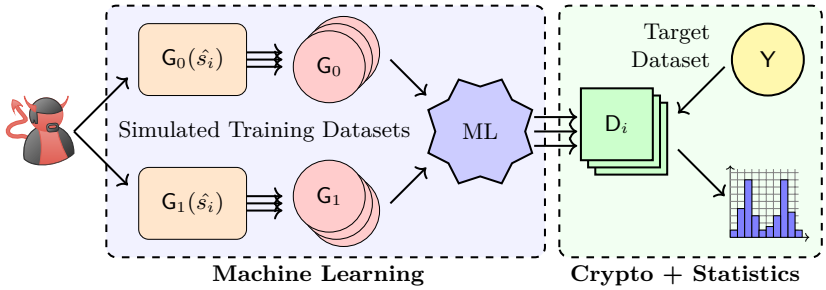


Figure 41: Abstract representation of our methodology.

Definition 33 (Distinguish problem)**.** *Let G_0 and G_1 be two classes, $b \leftarrow_R \{0,1\}$ a random coin-flip, and y an element of G_b. Consider a distinguisher D that takes as input y and outputs a guess b'. We define the* ***distinguish problem*** *as D's task in discriminating the membership of the value $y \in G_b$ between the two classes (G_0, G_1) and with advantage*[10]*:*

$$\mathsf{Adv}^{D}_{G_0,G_1} = \left| 2 \cdot Pr\left[D(y) = b_i\right] - 1 \right| \tag{20}$$

Even though the abstract definition form, the distinguishing problem appears as the core concept behind many important cryptographic security problems: *pseudorandomness* is defined as a distinguishing problem between a primitive G and a real random process; in an *indistinguishable cipher-plaintext attacks* it is required to distinguish a ciphertext between two possible messages, and; the *cipher suite problem* requires to discriminate between different primitives $(\mathsf{G}_0, \mathsf{G}_1)$.

3.1 Our Methodology: from Classifiers to Distinguishers

Our methodology, depicted in Fig. 41, is based on the idea that a supervised learning algorithm can be used by an adversary $\mathcal{A}$ to create a distinguisher D between two classes $(\mathsf{G}_0, \mathsf{G}_1)$. We must observe that a supervised learning algorithm requires an input of a *labelled dataset* of correctly classified values (y_i, G_{b_i}), such that $y_i \in \mathsf{G}_{b_i}$, that are used to define the classifier. Our methodology assumes that an adversary $\mathcal{A}$ can pre-compute any labelled *simulated* training dataset, *i.e.* $\mathcal{A}$ can easily compute *different but related* instances of $(\mathsf{G}_0, \mathsf{G}_1)$, *e.g.* by sampling a different secret key. In this way, $\mathcal{A}$ can simulate arbitrarily labelled datasets which might not refer to the original problem instance between $(\mathsf{G}_0, \mathsf{G}_1)$ but are somehow related and thus, we consider them as *correct*.

The output of the algorithm is a classifier D that works exactly as a distinguisher, *i.e.* provided an element y, it guesses whether y belongs to G_0 or G_1. The next step is to consistently evaluate the *accuracy* that this distinguisher obtains. For the sake of simplicity, in this paper, we consider the **classifier accuracy** as the probability of correctly guessing the class for every element of a **target dataset** Y. However, other mechanisms can also be used to evaluate the accuracy like computing the confusion matrix and cross-validate the obtained results. We formally[11] define the accuracy as:

$$\mathsf{Acc}^{\mathsf{D}}_{\mathsf{G}_0,\mathsf{G}_1}(\mathsf{Y}) = \Pr_{y_i \in \mathsf{Y}}\left[\mathsf{D}(y_i) = \mathsf{G}_{b_i}\right]$$

Observe that the distinguisher's accuracy highly depends on the target dataset Y. This implies that the accuracy computed by a distinguisher generated by our methodo-

[10] We omit to specify the classes, *i.e.* $\mathsf{Adv}^{\mathsf{D}}$, when they are clear by the context.
[11] We will omit to specify the classes whenever they are clear by the context.

logy **is not** directly related to the distinguisher's advantage previously described in the distinguishing problem of Def. 33. The reason is that the accuracy is computed over a target subset Y, which is generally much smaller than the set $\mathbb{Y}$ of all the possible elements. In other words, it is not possible to compare the accuracy $\Pr_{y_i \in \mathsf{Y}}[\mathsf{D}(y_i) = b_i]$ and the probability $\Pr_{y_i \in \mathbb{Y}}[\mathsf{D}(y_i) = b_i]$ because the target Y might not be *representative* of the whole space $\mathbb{Y}$, *i.e.* Y might, for example, only contain *"easy to classify"* elements providing therefore a high accuracy for D even though it might have no cryptographic advantage.

Roughly speaking, the accuracy can be seen as a statistical estimator of the advantage $\mathsf{Adv}^{\mathsf{D}}$ meaning that there is a strong conceptual gap between theoretical and empirical results. However, it is possible to estimate both the dimensions and the number of samples needed to achieve a **statistically relevant** distinguisher, *e.g.* by verifying some accuracy properties with an appropriate statistical test and later evaluate the *power analysis* to confirm/evaluate the amount of sample needed to reach statistic relevance.

For the rest of the paper, we assume that there is always a way to correctly generate statistically relevant distinguishers D_i for any pair of classes $(\mathsf{G}_0, \mathsf{G}_1)$. Furthermore, we refer to D's advantage as:

$$\mathsf{Adv}^{\mathsf{D}}_{\mathsf{G}_0,\mathsf{G}_1}(\mathsf{Y}) = \left| 2 \cdot \mathsf{Acc}^{\mathsf{D}}_{\mathsf{G}_0,\mathsf{G}_1}(\mathsf{Y}) - 1 \right|$$

Note that, whenever it is possible, the adversary $\mathcal{A}$ can generate many different training datasets, thus obtaining a set of n distinguishers $\{\mathsf{D}_i\}_{i=1}^{n}$ each having its own accuracy $\mathsf{Acc}^{\mathsf{D}_i}_{\mathsf{G}_0,\mathsf{G}_1}(\mathsf{Y})$. By correctly analysing the accuracy's distribution, $\mathcal{A}$ can consider different attack *strategies*. Let us explain this concept with an example. Suppose that all the distinguishers generated by $\mathcal{A}$ have the same accuracy of 0.5. This means that $\mathcal{A}$ has no advantage and therefore must abandon the idea of solving the distinguishing problem. Differently, if $\mathcal{A}$ observes that a distinguisher D_i has an accuracy $0.5 - \delta$ for some positive $\delta \in \mathbb{R}_+$, $\mathcal{A}$ can invert D_i's output to define a new distinguisher D'_i with accuracy $0.5 + \delta$. In this case, $\mathcal{A}$ can transform distinguishers with an advantage in making wrong guesses into distinguishers that make correct guesses with the same advantage.

In summary, our methodology allows an adversary $\mathcal{A}$ to produce ML generated distinguishers if $\mathcal{A}$ can: *(i)* pre-compute labelled simulated training datasets; *(ii)* obtain statistically relevant target datasets, and; *(iii)* run appropriate tests to evaluate the accuracy.

Consider an adversary $\mathcal{A}$ that, after executing our methodology, obtains several distinguishers of which she **does not** know the accuracy distribution. Despite the odd requirement, observe that this is the standard in practice since, to compute the accuracy distribution, it is required to obtain a *correct* target dataset which might not be obtainable, *e.g.* a primitive's security might be defined as a distinguisher problem where the adversary cannot query the correct primitive instantiation, thus not allowing $\mathcal{A}$ to get any target dataset.

The **blind spot paradox**, depicted in Fig. 42, is the paradoxical phenomenon where a *blind* adversary $\mathcal{A}$, that does not know whether a specific distinguisher has an advantage or not, is unable to *spot* how to correctly utilise the results, thus annihilating any advantage possessed. This paradox arises naturally whenever the accuracy is distributed symmetrically *w.r.t.* the probability of 0.5. Consider a distinguisher D and observe that, without any precise knowledge, it is impossible to know if D has a potential advantage δ or $-\delta$. The symmetric accuracy's distribution property implies that the probability of D being a *"good"* or a *"bad"* distinguisher is the same. For this reason, $\mathcal{A}$ is unable to properly utilise the potential advantage obtained, thus giving

rise to the paradox. To avoid the paradox, it is necessary to allow the adversary to receive *"hints"* in the form of a statistically relevant list of target's outputs correctly classified. In this way, the adversary can get an estimation of the accuracy distribution and use this information to *"filter out the bad"* distinguishers. This completely breaks the symmetry of the distinguishers and allows them to use the *"good"* distinguishers. Of course, these hints might not be allowed by some theoretical security's properties **but** might better represent a realistic usage of such property.

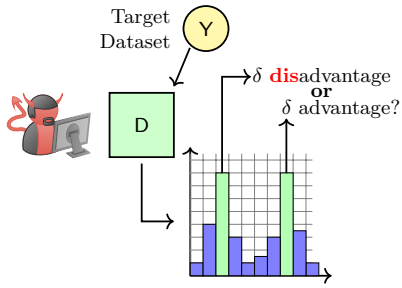


Figure 42: Representation of the blind spot paradox.

3.2 Distinguisher Accuracy Amplification

In this section, we propose a generalisation method to combine and amplify the advantage of several independent distinguishers into a more accurate one by assuming that all the distinguishers have the same accuracy. The underlying reasoning still holds even when considering different accuracy's distribution assumptions.

Let us assume we have n distinguishers $\{\mathsf{D}_i\}_{i=1}^n$, between classes $(\mathsf{G}_0, \mathsf{G}_1)$, all with the same accuracy $p > 0.5$. We require the distinguishers to be *independent* in the sense that they are generated from *different and independent training sets*. Our goal is to consider the *majority* of all the n distinguisher's guesses. In order to always have a majority, we must assume that n is odd, *i.e.* there exists $k \in \mathbb{N}$ such that $2k + 1 = n$.

Proposition 10. *Let $k \in \mathbb{N}$, $0.5 < p < 1$, $n = 2k + 1$ and $\{D_i\}_{i=1}^n$ be independent distinguishers with accuracy p. We define the distinguisher D' as the majority function of the n independent D_i guesses. Formally: $D'(y) = \mathsf{maj}(D_1(y), \ldots, D_n(y))$. Then, it holds that D' has an accuracy p_k greater than p.*

Proof. Note that the distinguisher' outputs define a binomial distribution of parameters p and n where the probability of *"t distinguishers are correct"* is:

$$\mathsf{Pr}\left[t \text{ are correct}\right] = \binom{2k+1}{t} \cdot (1-p)^{2k+1-t} \cdot p^t$$

The final guess of D' is defined by at least $k + 1$ distinguishers that have the same guess. This implies that the accuracy of D' directly depends on p and n. Formally, the probability of correctly guessing the distinguishing game for D', with $q = (1 - p)$, is:

$$\begin{aligned} p_k = \mathsf{Pr}\left[\mathsf{D}' \text{ correct}\right] = \mathsf{Pr}\left[\begin{matrix}\geq k+1 \\ \mathsf{D}_i \text{ correct}\end{matrix}\right] = \\ = \sum_{t=k+1}^{2k+1} \binom{2k+1}{t} \cdot q^{2k+1-t} \cdot p^t \end{aligned}$$

Let us recall the binomial identities $\binom{j}{k} = \binom{j-1}{k} + \binom{j-1}{k-1}$ and $\binom{2k-1}{t} = 0$ whenever $t > 2k-1$. Let us define p_0 to be exactly p. Our goal is to consider the probability p_k and obtain a relation *w.r.t.* p_{k-1}. Then, it holds that:

$$\begin{aligned}
p_k &= \sum_{t=k+1}^{2k+1} \binom{2k+1}{t} \cdot q^{2k+1-t} \cdot p^t \\
&= \sum_{t=k+1}^{2k+1} \left(\binom{2k-1}{t} + 2 \cdot \binom{2k-1}{t-1} + \binom{2k-1}{t-2} \right) \cdot q^{2k+1-t} \cdot p^t \\
&= \sum_{t=k+1}^{2k+1} \binom{2k-1}{t} \cdot q^{2k+1-t} \cdot p^t + \\
&\qquad + 2 \sum_{t=k+1}^{2k+1} \binom{2k-1}{t-1} \cdot q^{2k+1-t} \cdot p^t + \sum_{t=k+1}^{2k+1} \binom{2k-1}{t-2} \cdot q^{2k+1-t} \cdot p^t
\end{aligned} \tag{21}$$

Let us take a look at the addend and observe that it can be rewritten as:

$$\begin{aligned}
&\sum_{t=k+1}^{2k+1} \binom{2k-1}{t} \cdot q^{2k+1-p} \cdot p^t = q^2 \cdot \sum_{t=k+1}^{2k-1} \binom{2k-1}{t} \cdot q^{2k-1-t} \cdot p^t \\
&= q^2 \cdot \left(\sum_{t=k}^{2k-1} \binom{2k-1}{t} \cdot q^{2k-1-t} \cdot p^t \right) - q^2 \binom{2k-1}{k} \cdot q^{k-1} \cdot p^k \\
&= q^2 \cdot p_{k-1} - \binom{2k-1}{k} \cdot q^{k+1} \cdot p^k
\end{aligned} \tag{22}$$

where we note the presence of a relation to the winning probability p_{k-1}. Similarly, we manipulate the second and third addends and obtain:

$$\begin{aligned}
2 \sum_{t=k+1}^{2k+1} \binom{2k-1}{t-1} \cdot q^{2k+1-t} \cdot p^t &= 2 \cdot p \cdot q \cdot \sum_{t=k+1}^{2k} \binom{2k-1}{t-1} \cdot q^{2k-t} \cdot p^{t-1} \\
&= 2 \cdot p \cdot q \cdot \sum_{t=k}^{2k-1} \binom{2k-1}{t} \cdot q^{2k-1-t} \cdot p^t = 2 \cdot p \cdot q \cdot p_{k-1}
\end{aligned} \tag{23}$$

$$\begin{aligned}
\sum_{t=k+1}^{2k+1} \binom{2k-1}{t-2} \cdot q^{2k+1-t} \cdot p^t &= p^2 \cdot \sum_{t=k+1}^{2k+1} \binom{2k-1}{t-2} \cdot q^{2k+1-t} \cdot p^{t-2} \\
&= p^2 \cdot \sum_{t=k}^{2k-1} \binom{2k-1}{t} \cdot q^{2k-1-t} \cdot p^t + p^2 \cdot \binom{2k-1}{k-1} \cdot q^k \cdot p^{k-1} \\
&= p^2 \cdot p_{k-1} + \binom{2k-1}{k-1} \cdot q^k \cdot p^{k+1} = p^2 \cdot p_{k-1} + \binom{2k-1}{k} \cdot q^k \cdot p^{k+1}
\end{aligned} \tag{24}$$

where we used the fact that:

$$\binom{2k-1}{k-1} = \frac{(2k-1)!}{(k-1)! \cdot k!} = \binom{2k-1}{k}$$

By putting together Equations (22) to (24) into Eq. (21), it holds that:

$$\begin{aligned} p_k &= p_{k-1}\left(q^2 + 2qp + p^2\right) + \binom{2k-1}{k} \cdot q^k \cdot p^{k+1} - \binom{2k-1}{k} \cdot q^{k+1} \cdot p^k \\ &= p_{k-1}(q+p)^2 + \binom{2k-1}{k} \cdot (q \cdot p)^k \cdot (p-q) \\ &= p_{k-1} + \binom{2k-1}{k} \cdot (q \cdot p)^k \cdot (2p-1) \end{aligned}$$

from which we observe that $p_k > p_{k-1}$ whenever:

$$p_k > p_{k-1} \Leftrightarrow \binom{2k-1}{k} \cdot (q \cdot p)^k \cdot (2p-1) > 0 \Leftrightarrow (2p-1) > 0 \iff p > \frac{1}{2}$$

which is true by our hypothesis. The distinguisher D' built with $2k+1$ distinguisher has an accuracy $p_k > p_{k-1} > \cdots > p_0 = p$, concluding our proof. □

4 Case Study: Cipher Suite Distinguisher for Pseudorandom Generators

In this section, we implement our methodology into the MLCrypto tool which we use to create distinguishers for NIST DRBGs. We also discuss the connection between our empirical results and the constraints posed by a possible real attack against the primitives.

Let us consider a PRG $\mathsf{G} : \{0,1\}^{\ell_{\text{in}}} \to \{0,1\}^{\ell_{\text{out}}}$, as in Def. 31, and focus on the *pseudorandomness* property. Such a property states the indistinguishability between the distributions of the G's outputs and the uniformly random elements. By using the game-proving framework, it is required that any distinguisher D is unable to distinguish between a random value or G's output when provided by the challenger. Formally, we define the advantage as:

$$\mathsf{Adv}^{\mathsf{D}}_{\mathsf{G},\mathsf{rand}}(\lambda) = \left| \mathsf{Pr}\left[\mathsf{D}(\mathsf{G}(\mathsf{s})) = \mathsf{G}\right] - \mathsf{Pr}\left[\mathsf{D}(r) = \mathsf{G}\right] \right|$$

for some random seed $\mathsf{s} \leftarrow_R \{0,1\}^{\ell_{\text{in}}}$, uniformly sampled $r \leftarrow_R \{0,1\}^{\ell_{\text{out}}}$. The theoretical approach is conceptually simple and tight but infeasible because it requires a function that outputs random elements, which is, by other terms, precisely what the PRG tries to emulate, thus creating a brain-twisting loophole in which the goal is the solution at the same time.

To avoid this loophole, we can use a statistical approach, which consists of running several statistical tests using the outputs of G. After running G, the tests compare the real and the theoretical distributions to accept/reject the hypothesis that G is random or not. There are several statistical test suites to analyse the PRGs such as NIST STS [BRS+10], Dieharder [BEB13] and TestU01 [LS07].

Let us explain the approach with an example. Consider a list of N outputs $\{y_i\}_{i=1}^N$ from a pseudorandom G of which we want to determine if they appear random. To do so, consider the statistical test that shows the frequency of 1s in the output, *i.e.* it returns the number of 1s in a given output binary string.

Theoretically, we know that the output should describe the binomial distribution of which we know the *characteristic function*, *i.e.* the function that describes the probability distribution. For this reason, we apply the test on the set of outputs $\{y_i\}_{i=1}^N$ and

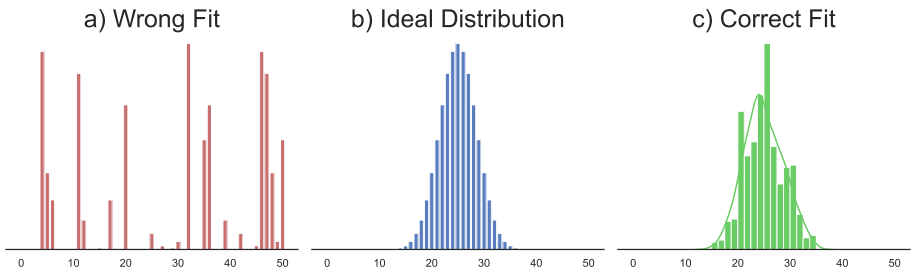


Figure 43: Example of distribution fitting *w.r.t.* an ideal binomial distribution.

compare it with the theoretical binomial ones thus testing if the outputs are *"binomial enough"*. In Fig. 43, we illustrate the possible outcomes of the test where we compare the ideal distribution *(b) w.r.t.* a fitting *(c)* and a completely random one *(a)*.

The tests take an analytical approach by computing precise values, *e.g.* the p-value for some specific statistical test. By repeating the test multiple times, it is possible to improve the *confidence* of the result. Sadly, regardless of the number of different tests we can perform and analyse, this approach can only state if a generator is plausibly pseudorandom or not.

On the other hand, the statistical approach allows the direct construction of a distinguisher D for the general pseudorandomness property, *i.e.* D executes the statistical test on the given output and uses the test results to discriminate between pseudorandom and non-pseudorandom. A failing test result, allows D to have an advantage in discriminate non-pseudorandom PRGs.

Let us take a step back and observe that the pseudorandom property can be modified into a *cipher suite distinguishing problem* in which a distinguisher D must distinguish between **two** different generators G_0 and G_1, regardless of their pseudorandom properties. By arithmetic manipulation of Eq. (20), we obtain:

$$
\begin{aligned}
\mathsf{Adv}^{\mathsf{D}}_{\mathsf{G}_0,\mathsf{G}_1}(\lambda) &= \left|\Pr\left[\mathsf{D}(\mathsf{G}_1(\mathsf{s}))=\mathsf{G}_1\right]-\Pr\left[\mathsf{D}(\mathsf{G}_0(\mathsf{s}))=\mathsf{G}_1\right]\right| \\
&= \left|\Pr\left[\mathsf{D}(\mathsf{G}_1(\mathsf{s}))=\mathsf{G}_1\right]-\Pr\left[\mathsf{D}(r)=\mathsf{G}_1\right]+\Pr\left[\mathsf{D}(r)=\mathsf{G}_1\right]-\Pr\left[\mathsf{D}(\mathsf{G}_0(\mathsf{s}))=\mathsf{G}_1\right]\right| \\
&\leq \left|\Pr\left[\mathsf{D}(\mathsf{G}_1(\mathsf{s}))=\mathsf{G}_1\right]-\Pr\left[\mathsf{D}(r)=\mathsf{G}_1\right]\right|+\left|\Pr\left[\mathsf{D}(r)=\mathsf{G}_1\right]-\Pr\left[\mathsf{D}(\mathsf{G}_0(\mathsf{s}))=\mathsf{G}_1\right]\right| \\
&\leq \mathsf{Adv}^{\mathsf{D}}_{\mathsf{G}_0,\mathsf{rand}}(\lambda)+\mathsf{Adv}^{\mathsf{D}}_{\mathsf{rand},\mathsf{G}_1}(\lambda)
\end{aligned}
$$

where the second addend:

$$
\left|\Pr\left[\mathsf{D}(r)=\mathsf{G}_1\right]-\Pr\left[\mathsf{D}(\mathsf{G}_0(\mathsf{s}))=\mathsf{G}_1\right]\right| \leq \mathsf{Adv}^{\mathsf{D}}_{\mathsf{rand},\mathsf{G}_1}(\lambda)
$$

measures the probability of D to wrongly distinguishing G_0. By the nature of the absolute value, we can modify this faulty distinguisher into a correct one by just flipping D's output. The idea behind our observation is that, by triangular disequality, distinguishing between two generators imposes a lower-bound on the generator's pseudorandomness advantage. Formally:

$$
\mathsf{Adv}^{\mathsf{D}}_{\mathsf{G}_0,\mathsf{G}_1}(\lambda) \leq \mathsf{Adv}^{\mathsf{D}}_{\mathsf{G}_0,\mathsf{rand}}(\lambda)+\mathsf{Adv}^{\mathsf{D}}_{\mathsf{rand},\mathsf{G}_1}(\lambda) \tag{25}
$$

Since executing the cryptanalysis necessary to create D is tedious, time-consuming and a human-intensive task, we use MLCrypto to automatically generate D from different NIST DRBG outputs.

4.1 Experiments and Results

In this section, we analyse the distinguishers generated by MLCrypto for the cipher suite distinguishing problem. Concretely, we focus on the DRBGs that NIST recommends [BK15]. Also, all the experiments we present in this section were run on an Intel(R) Core(TM) i7-4790 CPU @3.60GHz and 16GB of RAM with Linux. We implement MLCrypto in Python and all the source code of our tool is freely released for future research[12].

For this experiment, we consider the NIST DRBGs based on the primitives TDEA, AES-256, SHA-256 and HMAC-SHA-256. The choice of these DRBGs is arbitrary and if other primitives were chosen, the conclusions remain the same.

For all the experiments, there is a common initial phase where we calculate all possible pairs of combinations $(\mathsf{alg}_0, \mathsf{alg}_1)$ of the primitives and we accordingly generate the training and target datasets. For the training dataset, we want to simulate an adversary who cannot create such a dataset with the same seed as the target. Thus, all the training datasets have different seeds than the targets ones. In our case study, we analyse if the distribution of the accuracy of the distinguishers generated by MLCrypto (see Sec. 3) is affected by *(i)* the size of the datasets (training and target), and; *(ii)* different target dataset. The reason why we chose Naive Bayes classifiers for MLCrypto is that they are *(i)* computationally efficient, *(ii)* simple to implement, and; *(iii)* lack of learning parameter to be tuned.

Dataset size. To cross-validate our ML classifiers, we check if the size of the datasets affects the output of the distinguisher. To do so, we generate for each primitive alg a training dataset $\mathsf{X}_{\mathsf{alg}}$ containing n_{X} outputs of alg, and a target dataset $\mathsf{Y}_{\mathsf{alg}}$ containing n_{Y} outputs of alg. In more detail, the size of the training (n_{X}) and the target (n_{Y}) datasets are $n_{\mathsf{X}} \in \{2^i : i \in [12, 14]\}$ and $n_{\mathsf{Y}} \in \{2^i : i \in [14, 16]\}$ respectively. The datasets generation is computationally efficient and the size average with 2^{16} values is ~ 1.1 MB. We independently execute MLCrypto t_{X} times with a freshly generated training dataset, say X', **but** with the same target dataset Y. Concretely, we consider $t_{\mathsf{X}} = 2^{10}$ which would provide to compute a Cohen's coefficient of $d = 0.0876$ for a statistical power of $p = 0.8$, whenever analysing the distinguishers' accuracy distribution with a one-sample t-Student test with significance level $\alpha = 0.5$. In other words, the size of our datasets, as well as the number of tests, provide a (simplistic) statistical analysis that the obtained classifiers accuracy's distribution has some statistical confidence. In Fig. 44, we observe that changing the training dataset size n_{X} does not have any major impact on the accuracy distribution. This suggests that it is possible to provide smaller training datasets and still achieving the same accuracy distribution. Finally, we also checked our model's ability to predict new data (*i.e.* avoid overfitting or selection bias), we obtained the cross-validation value of each one of the experiments we performed. In more detail, we computed the 10-fold cross-validation using the function provided by *scikit-learn* and got a consistent accuracy in all our independent experiments.

Different targets. We generate the training datasets of such primitives and obtain a distinguisher D for the algorithms $(\mathsf{alg}_0, \mathsf{alg}_1)$. Once we have D, we randomly generate a target dataset and compute the accuracy of the distinguisher as $\mathsf{Acc}^{\mathsf{D}}_{\mathsf{alg}_0,\mathsf{alg}_1}$. Fig. 45 depicts that the same distinguishers define different accuracy distributions when computed on different target datasets. This phenomenon is explained by the fact that each target dataset is generated using a *different seed* thus making the generator *de facto* different. This implies that an increased accuracy advantage δ for a distinguisher D holds exclusively for a specific target. By changing the target, D changes the advantage to a different value δ'. We also consider a variation of n_{Y} and observe that the peaks

[12] https://bitbucket.org/CharlieTrip/mlcryptocode/src/master/

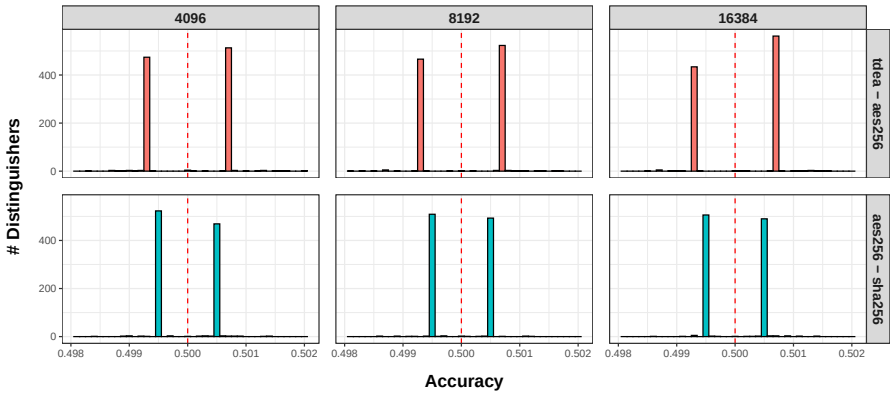


Figure 44: Distinguishers' accuracy distributions of two arbitrary primitives computed for 3 different training dataset sizes.

are differently spread. This is coherent when considering that a smaller dataset X' is a sample of a bigger one X, meaning that X' might not be a statistically significant representation of X. This implies the necessity of always using statistically significant target datasets when computing the accuracy distribution.

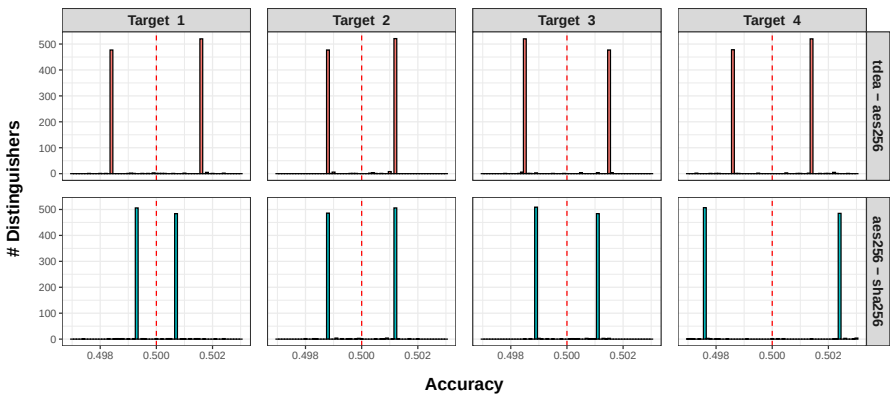


Figure 45: Distinguishers' accuracy distributions of two arbitrary primitives computed for 4 different target datasets.

Timing and space efficiency. In total, we generate $4 \cdot (1 + t_{\mathsf{X}}) = 4100$ independent datasets, being 4 the number of different primitives considered, and $\binom{4}{2} \cdot t_{\mathsf{X}} \cdot 3 = 18432$ distinguishers, being 3 the distinct n_{X} possible values. Each distinguisher outputs 3 values, being 3 the number of distinct n_{Y} possible values of a total of 55296 measurements. In Fig. 46 we show how the accuracy of the distinguishers is always distributed with either a single peak centred in 0.5 or as two symmetric peaks at value $0.5 \pm \delta$ for some non-negligible $\delta \in \mathbb{R}_+$ of the order of $\delta \sim 10^{-3}$. This demonstrates that MLCrypto can create a distinguisher D with advantage $\mathsf{Adv}^{\mathsf{D}} = 2\delta$. Even though that δ might initially be small when we consider only the distinguishers with accuracy $0.5 + \delta$, we apply the distinguisher' amplification method presented in Prop. 10 to increase up that advantage. For instance, in this case we have $\frac{t_{\mathsf{X}}}{2} - 1 = 511$ distinguishers with an accuracy of

$p \sim 50.1\%$ which implies that the amplification method creates a distinguisher D' with accuracy $p' \sim 51.8\%$.

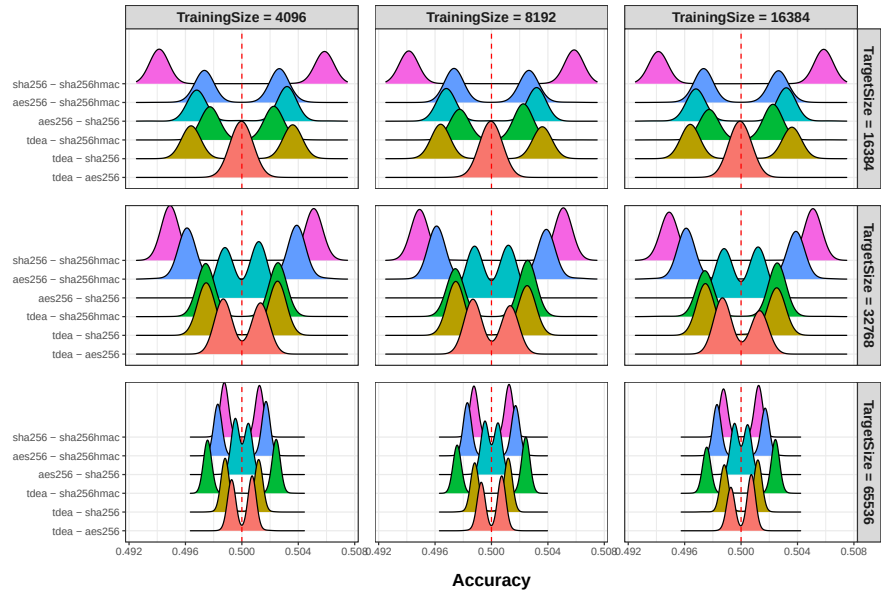


Figure 46: Distinguishers' accuracy distributions of the combination between the primitives in alg. We compute the distributions for 3 target dataset sizes and 3 training ones.

For completeness, we execute MLCrypto over all the NIST DRBGs, with training datasets size $n_{\mathsf{X}} = 2^{13}$ and target dataset size $n_{\mathsf{Y}} = 2^{16}$ of which accuracy distributions are depicted in Fig. 47. We observe that the accuracy distribution is always symmetric. This means that a blind adversary $\mathcal{A}$ must face the blind spot paradox, allowing us to empirically confirm that NIST DRBGs are, most probably, hard to distinguish between themselves. On the other hand, if $\mathcal{A}$ can reconstruct the distribution, then there is a concrete possibility to achieve a non-negligible advantage in distinguishing between the primitives.

5 Conclusions and Future Work

In this paper, we presented a methodology to use ML in developing practical distinguisher for cryptographic purposes. In particular, we show how it can be used for solving and analysing instances of distinguishing problems, *e.g.* we analyse the distinguishers obtained by MLCrypto for the cipher suite distinguishing problem between NIST DRBG. We foresaw the possibility of applying our tool to cipher suite distinguishing problems for block ciphers, hash functions, message authentication codes and similar primitives. The generality of our method allows it to be used for more practical problems related to *side-channel attacks* where the attacker is interested in distinguishing between two primitives based on non-cryptographic measurements, *e.g.* the power consumption and the computational timing and provides a consistent framework for future comparison between distinguishers generated by different ML approaches, *e.g.* random forest, neural network or the multi-layers perceptron model [BBCD20].

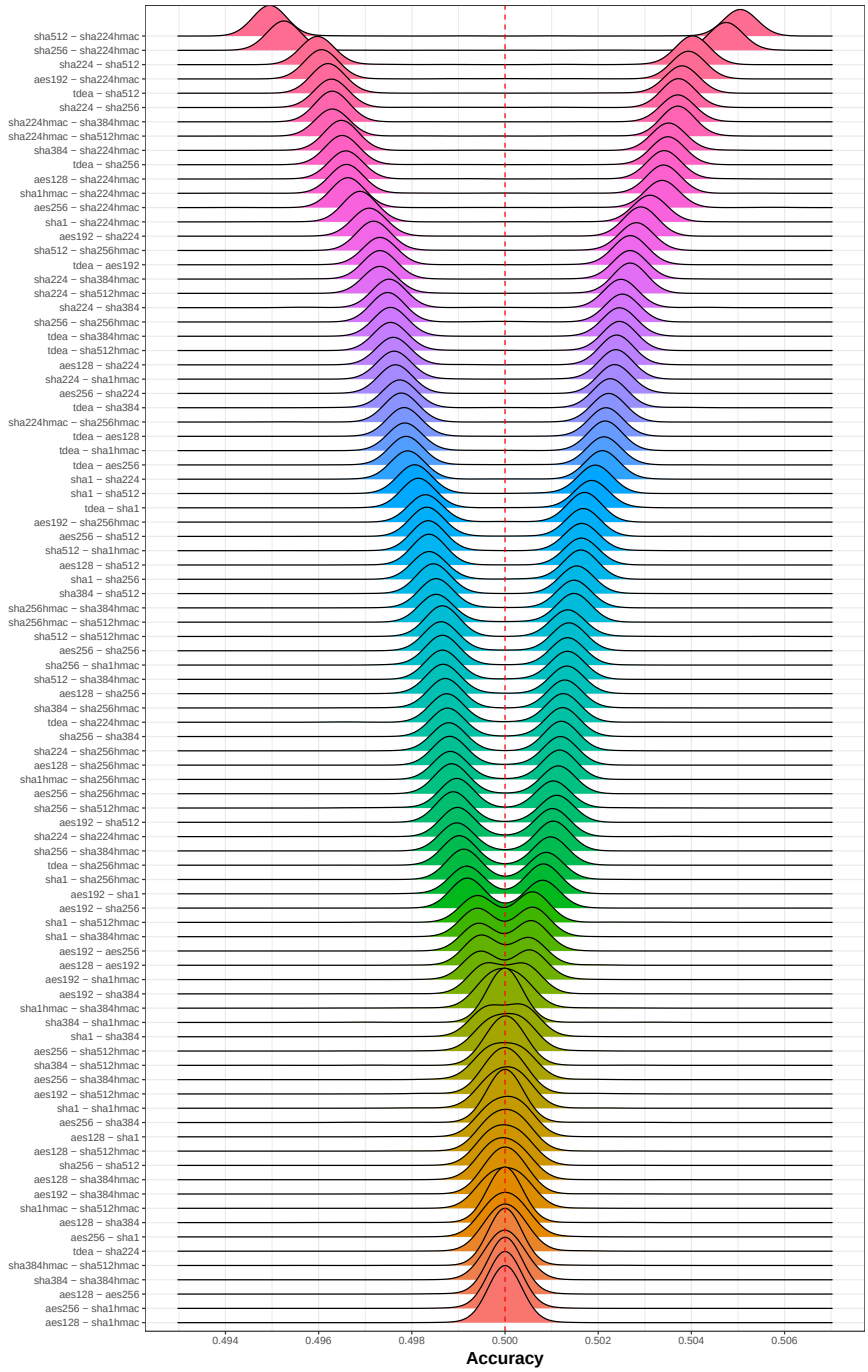


Figure 47: Distinguishers' accuracy distribution of all the NIST recommended DRBGs combinations with training dataset size $n_X = 2^{13}$ and target dataset size $n_Y = 2^{16}$.

Acknowledgements. This work was partially supported by the Swedish Foundation for Strategic Research (SSF).

Non-Interactive, Secure Verifiable Aggregation for Decentralized, Privacy-Preserving Learning

Carlo Brunetta[1], Georgia Tsaloli[1], Bei Liang[2], Gustavo Banegas[3] and Aikaterini Mitrokotsa[1,4]

[1] Chalmers University of Technology, Gothenburg, Sweden
[2] Beijing Institute of Mathematical Sciences and Applications, Beijing, China
[3] Inria and Laboratoire d'Informatique de l'Ecole polytechnique, Palaiseau, France
[4] University of St. Gallen, School of Computer Science, St. Gallen, Switzerland

26th Australasian Conference on Information Security and Privacy (ACISP), 2021, Perth (Australia)

Abstract: We propose a novel primitive called NIVA that allows the distributed aggregation of multiple users' secret inputs by multiple untrusted servers. The returned aggregation result can be publicly verified in a non-interactive way, *i.e.* the users are not required to participate in the aggregation except for providing their secret inputs. NIVA allows the secure computation of the sum of a large amount of users' data and can be employed, for example, in the federated learning setting in order to aggregate the model updates for a deep neural network. We implement NIVA and evaluate its communication and execution performance and compare it with the current state-of-the-art, *i.e.* Segal *et al.* protocol (CCS 2017) and Xu *et al.* VerifyNet protocol (IEEE TIFS 2020), resulting in better user's communicated data and execution time.

Keywords: Secure Aggregation, Privacy, Verifiability, Decentralization

1 Introduction

Smartphones, wearables and other Internet-of-Things (IoT) devices are all interconnected generating a lot of data, that often need to be aggregated to compute statistics in order to improve services. These improvements are often achieved by relying on *machine learning* (ML) algorithms, that simplify the prediction and/or inference of patterns from massive users' data. Given the high volume of data required, the ML paradigm creates serious privacy and security concerns [HAP17, LMA+18] that require a careful security analysis in order to guarantee the minimization of private information leakage while, concurrently, allowing the aggregation of the collected users' data. The growing storage and computational power of mobile devices as well as the increased privacy concerns associated with sharing private information, has led to a new distributed learning paradigm, *federated learning* [MMR+17] (FL). FL allows multiple users to collaboratively train learning models under the orchestration of a central server, while providing strong privacy guarantees by keeping the users' data stored on the source, *i.e.* the user's devices. More precisely, the central server collects and aggregates the local parameters from multiple users' and uses the aggregated value in order to train a global training model. The server plays the role of a central trusted **aggregator** that facilitates the communication between multiple users and guarantees the correct execution of the model update which, often, in current FL frameworks, is obtained by **summing** the individual users' parameters.

The shared model must be kept confidential since it might be employed to infer secret user information or disrupt the correct model update, *e.g.* a malicious server might bias the final result according to its preferences [HAP17, LMA+18, XEQ18, SS15, GGG17]. Furthermore, when the aggregation process is orchestrated by a single central server, this may lead to single *points-of-failure*. Our aim is to maximise the distributed nature of the learning process by: *(i) decentralizing* the aggregation process between multiple servers; *(ii)* providing the ability to *verify* the correctness of the *computed aggregation*; and *(iii)* guaranteeing the *confidentiality* of the users' inputs. Fig. 48 depicts the described scenario.

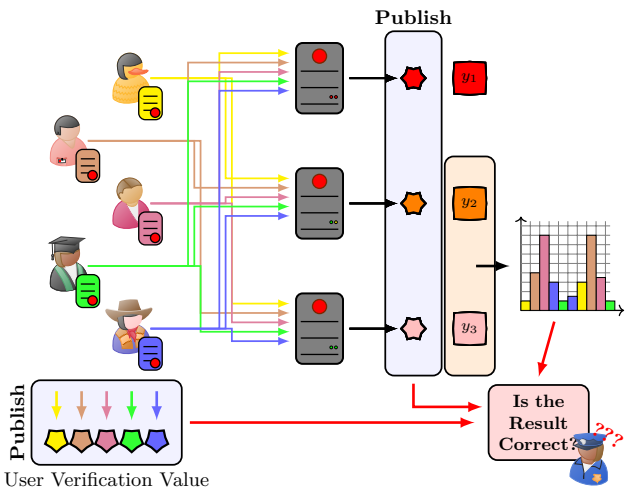


Figure 48: Several users delegate the secure aggregation of their inputs to independent servers. A threshold amount of server's outputs is necessary to publicly reconstruct and verify the resulting aggregated value.

1.1 Our Contributions

We define NIVA: a *Non-Interactive*, *decentralized* and publicly *Verifiable* secure *Aggregation* primitive inspired by the *verifiable homomorphic secret sharing* primitive introduced by Tsaloli *et al.* [TM20] **but** differs in both the construction and hypothesis. NIVA achieves decentralization by allowing the users to split their secret inputs and distribute the shares to multiple servers; while only a subset (*threshold*) of these servers need to collaborate in order to correctly reconstruct the output. Furthermore, NIVA allows the public verification of the computed aggregated value and contrary to existing work [BIK+17, XLL+20], NIVA is **non-interactive**, *i.e.* the users participate in the aggregation by releasing the appropriate messages and their participation is not required for the rest of the aggregation process. This allows NIVA to simplify the handling of *users' dropping out* from the aggregation process, which is a complex problem to handle in the case of interactive protocols. We further discuss possible optimization to the verification algorithm and extensions useful for realistic application, *e.g.* verification of users' shares, multiple executions and how to introduce a differentially private [Dwo06] mechanism. We implement NIVA, evaluate the communication costs, execution time, and perform a detailed experimental analysis. Furthermore, we compare our primitive with the current state-of-the-art, *i.e.* the secure aggregation protocols PPML and VerifyNet proposed by Segal *et al.* [BIK+17] and Xu *et al.* [XLL+20] correspondingly. NIVA optimizes the users' output and execution time making it several order of magnitude more suitable than PPML and VerifyNet for the FL setting that requires a big amount of users, *i.e.* more than 10^5 users.

1.2 Related Work

This work addresses a general problem that lies in the intersection of *"decentralized aggregation"* and *"verifiable delegation of computations"*.

Secret Sharing. A *threshold* secret sharing (SS) scheme allows a user to split a secret x into multiple shares $(x_1, x_2, \ldots, x_m)$ that are distributed to different servers. Whenever at least a *threshold* number t of servers collaborates by exchanging their shares, they are able to reconstruct the original secret. If any malicious adversary controls less than this threshold, it is impossible to reconstruct x. The first instantiation was provided by Shamir [Sha79]. In the following decades, several publications [Bei11, Bri90, GK06, Kra94] expanded Shamir's concept by providing schemes with additional properties such as *verification* and *homomorphism*.

An *additive homomorphic secret sharing* (additive HSS) allows the server to aggregate several shares coming from different users into a single one which, when correctly reconstructed, will allow the reconstruction of the *sum* of the original secrets. Besides Shamir's, the first instance of such a scheme was proposed by Benealoh [Ben87] and many other variations can be found in the literature [BGI17, FGJS17, LMS18].

Generally, the *verifiability* property describes the possibility to verify that some specific value is *"correctly evaluated"*. Whenever considering this property in the context of SS, it must be specified if *(a)* the server wants to verify the user's received shares; or *(b)* anyone wants to check if the servers' reconstructed secret is indeed the correct one. Chor *et al.* [CGMA85] provided the first SS scheme that is able to identify the existence of a *"cheating"* user, while Stadler [Sta96] extended it in order to detect both cheating users and servers. Tsaloli *et al.* [TLM18] proposed a *verifiable homomorphic secret sharing* (VHSS) scheme in which the *verifiability* property holds by *assuming the user's honesty* in generating the shares **but** allows the verification of the server's aggregation correctness. In this paper, we consider the properties of *verifiability* and *homomorphic secret sharing* as considered by Tsaloli *et al.* [TLM18]. Our primitive NIVA is *inspired*

by Tsaloli *et al.*'s primitive [TM20], *however*, it is based on a completely different construction.

Federated Learning and Cryptography. The setting posed by federated learning (FL) is similar to the aggregation problems we consider. Concretely, every time the FL model must be updated, the users send their parameters to the server that must provide the final aggregated model back. The work in Bonawitz *et al.* [BIK+17] proposes a secure aggregation protocol, called PPML, that achieves security and privacy with a major focus on maintaining high efficiency. This solution provides a procedure to correctly handle *users' drop-outs*, *i.e.* users that are unable to correctly terminate the protocol. In the same spirit, Xu *et al.* [XLL+20] introduced VerifyNet, an (conceptually) extended version of PPML that introduces a *public verification* procedure to check the correctness of the aggregation process. However, these solutions are based on a single central server, and they are therefore susceptible to *single point-of-failure*, *i.e.* if the central server crashes, the whole protocol aborts. To avoid this, it is required to *distribute/decentralise* the role of the central server, *e.g.* by either introducing *threshold* cryptographic primitives between multiple aggregators [THH+09] or completely decentralising the aggregation using a blockchain [CZD+19]. Recently, privacy-preserving aggregation problems have gained substantial attention in the past few years [TB19, GGG17, XLL+20, CZD+19, THH+09, SS15, PAH+18, BIK+17]. The solutions presented achieve different properties related to security, privacy, and verifiability by considering specific cryptographic assumptions, security models, and/or application requirements. Our primitive allows to *publicly verify* the correctness of the final output, handles the users' drop-outs **as well as** possible servers' failure by distributing the aggregation computation among several *independent* servers.

1.3 Paper Organisation

Sec. 2 contains the necessary preliminaries used throughout the paper. Sec. 3 introduces our primitive NIVA, its security and verifiability properties, further discusses additional properties and compares to the related work. Sec. 4 describes NIVA's implementation details and showcases relevant performance statistics, *e.g.* execution timing and bandwidth usage in relation to scaling the amount of users and servers. Furthermore, we compare our implementation with Segal *et al.* [BIK+17] and Xu *et al.* [XLL+20] for similar evaluation parameters.

2 Preliminaries

In this section, we show the definitions used throughout the paper.

Denote with $\mathsf{Pr}[E]$ the probability that the event E occurs. Let the natural number be denoted by $\mathbb{N}$, the integer number ring with $\mathbb{Z}$, the real number field with $\mathbb{R}$ and the positive ones with $\mathbb{R}_+$. Let $[a, b]$ denote intervals between a and b. Let $|X| \in \mathbb{N}$ indicate the cardinality of the set X and $\mathsf{rk}(A)$ the rank of the matrix A. Let $\sum_{x\in X}^{y\in Y}$ be the sum $\sum_{x\in X, y\in Y}$, respectively $\prod_{x\in X}^{y\in Y}$ is $\prod_{x\in X, y\in Y}$.

Theorem 8 (Rouché-Capelli [CG86])**.** *An n-variable linear equation system $Ax = b$ has a solution $\Leftrightarrow$ $\mathsf{rk}(A) = \mathsf{rk}(A|b)$ where $A|b$ is the augmented matrix,* i.e. *A with appended the column b.*

Key Agreement. Let $\mathbb{G}$ be a cyclic group of order p prime with generator g, *e.g.* groups based on elliptic curves [Kob87]. Let us report the Diffie-Hellman key agreement [DH76] and the related assumptions.

Assumption 5 (Diffie-Hellman Assumptions)**.** *Consider a cyclic group $\mathbb{G}$ of prime order p with generator g and $a, b \in [0, p-1]$. Given elements $(A, B) = (g^a, g^b)$, the **computation Diffie-Hellman problem (CDH)** requires to compute the element $g^{ab} \in \mathbb{G}$. The **decisional Diffie-Hellman problem (DDH)** requires to correctly distinguish between (g, A, B, g^{ab}) and (g, A, B, g^c) for some random $c \in [0, p-1]$. We assume the advantage of solving the CDH and the DDH problems to be negligible,* i.e. *$\epsilon_{CDH} < \mathsf{negl}$ and $\epsilon_{DDH} < \mathsf{negl}$.*

Definition 34 (Diffie-Hellman Key Exchange)**.** *The Diffie-Hellman key agreement scheme is defined with the following algorithms:*

- *$\mathsf{KSetup}(\lambda) \rightarrow pp$: the setup algorithm takes as input the security parameter and outputs the public parameters pp which contains a prime p, the description of a cyclic group $\mathbb{G}$ of order p and generator g for the group $\mathbb{G}$.*
- *$\mathsf{KGen}(pp) \rightarrow (\mathsf{sk}, \mathsf{pk})$: the key generation algorithm samples the secret key $\mathsf{sk} \in [0, p-1]$ and computes the public key $\mathsf{pk} = g^{\mathsf{sk}}$. It outputs $(\mathsf{sk}, \mathsf{pk}) = (\mathsf{sk}, g^{\mathsf{sk}})$.*
- *$\mathsf{KAgree}(sk_i, pk_j) \rightarrow s_{ij}$: the key agreement algorithm takes in input a secret key sk_i and a public key $pk_j = g^{sk_j}$ and outputs the shared secret $s_{ij} = pk_j^{sk_i} = g^{sk_j \cdot sk_i}$.*

*The scheme is said to be **correct** if for any $pp \leftarrow \mathsf{KSetup}(\lambda)$, $(sk_i, pk_i) \leftarrow \mathsf{KGen}(pp)$ and $(sk_j, pk_j) \leftarrow \mathsf{KGen}(pp)$, it holds that $\mathsf{KAgree}(sk_i, pk_j) = s_{ij} = s_{ji} = \mathsf{KAgree}(sk_j, pk_i)$. The scheme is said to be **secure** if for any $pp \leftarrow \mathsf{KSetup}(\lambda)$, and keys $(sk_i, pk_i) \leftarrow \mathsf{KGen}(\mathsf{pp}, \mathcal{U}_i)$, $(sk_j, pk_j) \leftarrow \mathsf{KGen}(\mathsf{pp}, \mathcal{U}_j)$, it holds that any PPT adversary $\mathcal{A}$ has negligible probability to compute s_{ij} from (pk_i, pk_j) which reduces to the CDH Assumption 5.*

For our primitive, we use the shared secret s_{ij} as a pseudorandom integer despite being an element of the group $\mathbb{G}$. This is possible by considering a generic hash function H mapping the group $\mathbb{G}$ to the integers $\mathbb{Z}$, which translates s_{ij} into a *pseudorandom* integer. To avoid heavy notation, we denote this output as s_{ij}.

Additionally, consider the *discrete logarithm problem* for a subset I, *i.e.* the dLog problem where the solution is contained in a subset $I \subseteq [0, p-1]$.

Assumption 6 (Discrete Logarithm in Subset I Problem)**.** *Consider $\mathbb{G}$ a cyclic group of prime order p with generator g and a subset $I \subseteq [0, p-1]$. Given $y \in \mathbb{G}$, the **discrete logarithm problem for the subset I (dLog_I)** requires to find the value $x \in I$ such that $g^x = y$.*

In order to assume the dLog_I problem to be computationally hard, the cardinality of I needs to be *"big enough"*, *i.e.* if $|I| > 2^{160}$ then the kangaroo Pollard's rho algorithm [Pol00] has complexity $\sim 2^{80}$ which we consider to be infeasible.

Secret Sharing. We report the additive homomorphic SS scheme's definition.

Definition 35 (Additive Homomorphic SS Scheme)**.** *Let $n, m, t \in \mathbb{N}$ such that $0 < t < m$. For each $i \in [1, n]$, let $x_i \in \mathbb{F}$ be the secret input of the user $\mathcal{U}_i$ for some input space $\mathbb{F}$. Consider the set of servers $\mathsf{S} = \{\mathcal{S}_j\}_{j \in [1,m]}$. Define (t, m)-**threshold additive homomorphic secret sharing** scheme as:*

- *$\mathsf{SS.Share}(x_i, t, \mathsf{S}) \rightarrow \{x_{ij}\}_{j \in [1,m]}$: given the secret input x_i, the threshold value t and the list of servers S, the share generation algorithm outputs a list of m shares x_{ij} for $j \in [1, m]$, one for each server $\mathcal{S}_j$.*
- *$\mathsf{SS.Eval}(\{x_{ij}\}_{i \in [1,n]}) \rightarrow y_j$: given as input a set of shares x_{ij} for the same server $\mathcal{S}_j$, the evaluation algorithm outputs an aggregated share y_j.*
- *$\mathsf{SS.Recon}(t, \{y_j\}_{j \in \mathcal{T}}) \rightarrow y$: given as input the threshold value t and a list of shares y_j for a subset of servers $\mathcal{S}_j \in \mathcal{T} \subseteq \mathsf{S}$ such that $|\mathcal{T}| > t$, the reconstruction algorithm outputs the reconstructed secret y.*

A (t, m) additive homomorphic secret sharing scheme is said to be ***correct*** *if for all $i \in [1, n]$, any choice of secrets $x_i \in \mathbb{F}$, for all the shares $\mathsf{SS.Share}(x_i, t, \mathsf{S}) \rightarrow \{x_{ij}\}_{j\in[1,m]}$, aggregated shares $\mathsf{SS.Eval}(\{x_{ij}\}_{i\in[1,n]}) \rightarrow y_j$, for all the servers' reconstruction subset $\mathcal{T}$ such that $|\mathcal{T}| > t$, it holds that the reconstructed value $\mathsf{SS.Recon}(t, \{y_j\}_{j\in\mathcal{T}}) \rightarrow y$ is equal to $y = \sum_{i=1}^{n} x_i$.*

A (t, m) additive homomorphic secret sharing scheme is ***secure*** *if for all $i \in [1, n]$, any secrets $x_i \in \mathbb{F}$, for all the shares $\mathsf{SS.Share}(x_i, t, \mathsf{S}) \rightarrow \{x_{ij}\}_{j\in[1,m]}$, aggregated shares $\mathsf{SS.Eval}(\{x_{ij}\}_{i\in[1,n]}) \rightarrow y_j$, an adversary $\mathcal{A}$ that controls a servers' subset $\mathcal{T} \subseteq \mathsf{S}$, such that $|\mathcal{T}| \leq t$, is unable to obtain the reconstructed value y.*

3 NIVA

In this section, we describe the decentralised aggregation problem's setting as well as the security and privacy requirements and how they must guarantee public verifiability of the aggregated computations. We instantiate NIVA and define the security and verifiability properties.

Consider n users $\mathcal{U}_i$, each owns a secret input x_i, and m servers $\mathcal{S}_j$. The goal is to *distribute the computation* of the sum of the users secret inputs' $\sum_{i=1}^{n} x_i$ between the m servers of which only a *designed threshold* amount $t + 1 \leq m$ of servers is required to obtain the aggregated value. Formally:

Definition 36. *Let the algorithms* $(\mathsf{Setup}, \mathsf{SGen}, \mathsf{Agg}, \mathsf{Ver})$ *defined as:*

- $\mathsf{Setup}(\lambda) \rightarrow (\mathsf{sk}_I, \mathsf{pk}_I)$ *: given the security parameter λ, the setup algorithm provides a keypair $(\mathsf{sk}_I, \mathsf{pk}_I)$ associated to the user/server I.*
- $\mathsf{SGen}(x_i, \mathsf{sk}_{\mathcal{U}_i}, t, \{\mathsf{pk}_{\mathcal{S}_j}\}_{j=1}^{m}) \rightarrow (\mathsf{pk}_{\mathcal{U}_i}, \{\widehat{x_{ij}}\}_{j=1}^{m}, R_i, \{\tau_{ij}\}_{j=1}^{m})$ *: given a secret input $x_i \in I$ and the user's $\mathcal{U}_i$ secret key $\mathsf{sk}_{\mathcal{U}_i}$, the designed threshold amount $0 < t < m-1$ and the list of servers' public keys $\{\mathsf{pk}_{\mathcal{S}_j}\}_{j=1}^{m}$ from which we obtain the list of servers' identities $\{\mathcal{S}_j\}_{j=1}^{m}$, the share generation algorithm outputs the shares $\widehat{x_{ij}}$, additional information R_i and the verification coefficients τ_{ij} to be either shared with the server $\mathcal{S}_j$ or publicly released.*
- $\mathsf{Agg}(\{(\mathsf{pk}_{\mathcal{U}_i}, \widehat{x_{ij}}, R_i)\}_{i\in N}, \mathsf{sk}_{\mathcal{S}_j}) \rightarrow (y_j, \pi_j, R_{\mathcal{S}_j}, \rho_j)$ *: given a set of public keys, shares and additional information $(\widehat{x_{ij}}, R_i, \mathsf{pk}_i)$ for a list of users $\mathcal{U}_i$ in the subset $N \subseteq [1, n]$, the aggregation algorithm outputs the partial evaluation y_j, a partial verification proof π_j and additional information $(R_{\mathcal{S}_j}, \rho_j)$.*
- $\mathsf{Ver}\Big(t, \{\tau_{ij}\}_{i\in N}^{j\in M}, \{(y_j, \pi_j, R_{\mathcal{S}_j}, \rho_j)\}_{j\in M}\Big) \rightarrow \{y, \perp\}$ *: given the threshold t, a set of servers M with $t+1 \leq |M| \leq m$, given partial evaluations, proofs and additional information $(y_j, \pi_j, R_{\mathcal{S}_j}, \rho_j)$ and a set of verification coefficients $\{\tau_{ij}\}_{i\in N}^{j\in M}$ for a subset of users N, the verification algorithm outputs the aggregated value $y = \sum_{i\in N} x_i$ if the servers correctly computed the aggregation of their shares. Otherwise, outputs $\perp$.*

The primitive must be **correct**, *i.e.* the verification always outputs $y = \sum_{i\in N} x_i$ whenever using correctly aggregated outputs computed from correctly generated shares of the secrets $\{x_i\}_{i\in N}$. Additionally, the users' input must be **secure**. The security experiment describes a realistic scenario in which the adversary $\mathcal{A}$ must recover the secret inputs x_i, which are randomly sampled by the challenger $\mathcal{C}$. The amount of servers that $\mathcal{A}$ is able to compromise is at most t since this servers' subset is not enough for using the secret share's reconstruction algorithm $\mathsf{SS.Recon}$. Our experiment includes the *single-user input privacy*, *i.e.* whenever $\mathcal{A}$ requests a challenge for $n = 1$, the property holds for the input x_i.

Definition 37 (Security)**.** *Consider the primitive of Def. 36 be defined between n users and m servers and threshold t. Let $\mathcal{A}$ be a PPT adversary that maliciously controls t servers, w.l.o.g. $\{\mathcal{S}_j\}_{j=1}^{t}$. Consider the security experiment $\mathsf{Exp}^{\mathsf{sec}}(\mathcal{A})$:*

1. *For every $j \in [1,t]$, the challenger $\mathcal{C}$ executes $\mathsf{Setup}(\lambda)$ and sends to $\mathcal{A}$ all the corrupted servers' key-pairs $(\mathsf{sk}_{\mathcal{S}_j}, \mathsf{pk}_{\mathcal{S}_j})$, while for the remaining $j \in [t+1, m]$ servers, it returns only the non-corrupted server's public key $\mathsf{pk}_{\mathcal{S}_j}$.*
2. *$\mathcal{A}$ outputs to $\mathcal{C}$ the number of users n to be challenged on.*
3. *$\mathcal{C}$ executes $\mathsf{Setup}(\lambda)$ and generates the key pairs $(\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i})$ and randomly samples an input $x_i \in I$ for each user $\mathcal{U}_i$.*
4. *$\mathcal{C}$ computes the shares $\mathsf{SGen}\big(x_i, \mathsf{sk}_{\mathcal{U}_i}, t, \{\mathsf{pk}_{\mathcal{S}_j}\}_{j=1}^{m}\big)$ and outputs to $\mathcal{A}$ the compromised servers' shares $\big(\mathsf{pk}_{\mathcal{U}_i}, \{\widehat{x_{ij}}\}_{j=1}^{t}, R_i\big)$ plus all the verification values $\{\tau_{ij}\}_{j=1}^{m}$ for each $i \in [1,n]$.*
5. *$\mathcal{A}$ outputs the aggregated secret $y^\star$.*
6. *If $y^\star = \sum_{i=1}^{n} x_i$, the experiment outputs 1, otherwise 0.*

*The primitive is said to be **secure** if $\Pr[\mathsf{Exp}^{\mathsf{sec}}(\mathcal{A}) = 1] < \mathsf{negl}$.*

Finally, we require to **publicly verify** the computations of the servers, *i.e.* the servers must provide a proof of the correct computation. In other words, the verifiability property requires the impossibility for an adversary $\mathcal{A}$ to force the correct verification of a wrong aggregated value. This property holds whenever there exists at least one honestly computed partial evaluation, regardless of the number of servers that $\mathcal{A}$ compromises. On the other hand, whenever $\mathcal{A}$ controls more than t servers, the security property does not hold, thus obtaining a potentially verifiable primitive but definitely not secure. For this reason, we design the verifiability experiment in which, before obtaining the correct partial evaluations, $\mathcal{A}$ is allowed to select the subset of inputs $N^\star$ to be aggregated and, after receiving the non-compromised partial evaluations, $\mathcal{A}$ outputs tampered partial evaluations for the compromised servers and selects a set $M^\star$ of evaluations to be used in the verification challenge. The adversarial set $M^\star$ must contain at least one honestly generated partial evaluation and it is used to describe the realistic attack scenario in which the adversary denies the verifier to obtain all the partial evaluations **but** at least a honest one is present.

Definition 38 (Verifiability)**.** *Consider the primitive of Def. 36 defined between n users, m servers and threshold t. Let $\mathcal{A}$ be a PPT adversary that maliciously controls $k < m$ servers, w.l.o.g. $\{\mathcal{S}_j\}_{j=1}^{k}$. Consider the experiment $\mathsf{Exp}^{\mathsf{ver}}(\mathcal{A})$:*

1. *For every $j \in [1,k]$, the challenger $\mathcal{C}$ executes $\mathsf{Setup}(\lambda)$ and sends to $\mathcal{A}$ all the corrupted servers' key-pairs $(\mathsf{sk}_{\mathcal{S}_j}, \mathsf{pk}_{\mathcal{S}_j})$, while for the remaining $j \in [k+1, m]$ servers it returns only the non-corrupted server's public key $\mathsf{pk}_{\mathcal{S}_j}$.*
2. *$\mathcal{A}$ outputs to $\mathcal{C}$ the number of users n to be challenged on.*
3. *$\mathcal{C}$ executes $\mathsf{Setup}(\lambda)$ and generates the key pairs $(\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i})$ and randomly samples an input $x_i \in I$ for each user $\mathcal{U}_i$.*
4. *$\mathcal{C}$ computes the shares $\mathsf{SGen}\big(x_i, \mathsf{sk}_{\mathcal{U}_i}, t, \{\mathsf{pk}_{\mathcal{S}_j}\}_{j=1}^{m}\big)$ and outputs to $\mathcal{A}$ the compromised servers' shares $\big(\mathsf{pk}_{\mathcal{U}_i}, \{\widehat{x_{ij}}\}_{j=1}^{k}, R_i\big)$ plus all the verification values $\{\tau_{ij}\}_{j=1}^{m}$ for each $i \in [1,n]$.*
5. *$\mathcal{A}$ provides to $\mathcal{C}$ the list of inputs $N^\star$ to be challenged.*

6. *For each non compromised server $\mathcal{S}_j$ where $j \in [k+1, m]$, $\mathcal{C}$ returns to $\mathcal{A}$ the $\mathcal{S}_j$'s partial evaluations* $(y_j, \pi_j, R_{\mathcal{S}_j}, \rho_j) \leftarrow \mathsf{Agg}\big(\{(\mathsf{pk}_{\mathcal{U}_i}, \widehat{x_{ij}}, R_i)\}_{i\in N^\star}, \mathsf{sk}_{\mathcal{S}_j}\big)$.

7. *$\mathcal{A}$ outputs tampered evaluations* $\{y_j{}^\star, \pi_j{}^\star, R_{\mathcal{S}_j}{}^\star, \rho_j{}^\star\}_{j=1}^{k}$.

8. *$\mathcal{A}$ provides to $\mathcal{C}$ the list of verifying servers $M^\star$ in which there exists a non-compromised server $\mathcal{S}_l \in M^\star$ with $l \in [k+1, m]$.*

9. *The experiment computes the verification algorithm*

$$\mathsf{Ver}\Big(t, \{\tau_{ij}\}_{i\in N^\star}^{j\in M^\star}, \big\{(y_j, \pi_j, R_{\mathcal{S}_j}, \rho_j)\big\}_{j\in M^\star}\Big) \rightarrow y^\star$$

and outputs 1 if $y^\star \neq y = \sum_{i\in N^\star} x_i$, otherwise 0.

The primitive is said to be ***verifiable*** *if* $Pr[Exp^{ver}(\mathcal{A}) = 1] < \mathsf{negl}$.

3.1 NIVA Instantiation

In this section, we provide our instantiation of Def. 36, called NIVA. In a nutshell, NIVA incorporates into the Shamir's SS scheme of Sec. 2, the usage of a key-agreement scheme between the users and the servers. This allows the creation of a *"proving value"* used during the verification phase which **must** be correctly computed by the servers or, otherwise, the verification process fails.

Definition 39 (NIVA)**.** *Let* $(\mathsf{KSetup}, \mathsf{KGen}, \mathsf{KAgree})$ *be a key agreement (Def. 34) with public parameters* $\mathsf{pp} \leftarrow \mathsf{KSetup}(\lambda)$*, defined over a cyclic group $\mathbb{G}$ with prime order p. Let $n \in \mathbb{N}$ be the number of users $\mathcal{U}_i$ and $m \in \mathbb{N}$ be the number of servers $\mathcal{S}_j$. Let I be a secret input's space closed under summation such that the dLog_I problem of Assumption 6 is hard. Let $N \subseteq [1, n]$ be a users' subset and $M \subseteq [1, m]$ a servers' subset. We refer to $\mathcal{S}_j \in M$ with $j \in M$. Let $t \in \mathbb{N}$ be the evaluation threshold such that $0 < t < m$. Define NIVA with algorithms:*

- $\mathsf{Setup}(\lambda) \rightarrow (\mathsf{sk}_I, \mathsf{pk}_I)$ *: given the security parameter λ, the setup algorithm executes* $\mathsf{KGen}(\mathsf{pp})$ *and outputs the result* $(\mathsf{sk}_I, \mathsf{pk}_I) = \big(\mathsf{sk}_I, g^{\mathsf{sk}_I}\big)$*. The* Setup *algorithm is evaluated by each user $\mathcal{U}_i$ and server $\mathcal{S}_j$. All the public keys of the servers* $\{\mathsf{pk}_{\mathcal{S}_j}\}_{j=1}^{m}$ *are publicly released.*

- $\mathsf{SGen}\big(x_i, \mathsf{sk}_{\mathcal{U}_i}, t, \{\mathsf{pk}_{\mathcal{S}_j}\}_{j=1}^{m}\big) \rightarrow \big(\mathsf{pk}_{\mathcal{U}_i}, \{\widehat{x_{ij}}\}_{j=1}^{m}, R_i, \{\tau_{ij}\}_{j=1}^{m}\big)$ *: given a secret input $x_i \in I$ and the user's $\mathcal{U}_i$ secret key* $\mathsf{sk}_{\mathcal{U}_i}$*, the designed threshold amount $0 < t < m-1$, the list of servers' public keys* $\{\mathsf{pk}_{\mathcal{S}_j}\}_{j=1}^{m}$ *from which we obtain the list of servers' identities* $\{\mathcal{S}_j\}_{j=1}^{m}$*, the share generation algorithm instantiates a (t, m)-threshold additive homomorphic secret sharing scheme by executing the algorithm* $\mathsf{SS.Share}\big(x_i, t, \{\mathcal{S}_j\}_{j=1}^{m}\big)$ *which returns the shares $\widehat{x_{ij}}$ for all $j \in [1, m]$. Then, $\mathcal{U}_i$ uses its secret key* $\mathsf{sk}_{\mathcal{U}_i}$ *to compute the shared secrets* w.r.t. *each server $\mathcal{S}_j$,* i.e. $\mathsf{KAgree}(\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{S}_j}) \rightarrow s_{ij}$*. The algorithm samples a random value $r_i \in [0, p-1]$, computes $R_i = g^{r_i}$, and the verification coefficients*

$$\tau_{ij} = \mathsf{pk}_{\mathcal{S}_j}^{x_i} \cdot R_i^{s_{ij}} = g^{\mathsf{sk}_{\mathcal{S}_j} x_i + r_i \cdot s_{ij}} \tag{26}$$

The algorithm outputs $\big(\mathsf{pk}_{\mathcal{U}_i}, \{\widehat{x_{ij}}\}_{j=1}^{m}, R_i, \{\tau_{ij}\}_{j=1}^{m}\big)$*. Each user publicly releases the values* $\{\tau_{ij}\}_{j=1}^{m}$*.*

- $\mathsf{Agg}\big(\{(\mathsf{pk}_{\mathcal{U}_i}, \widehat{x_{ij}}, R_i)\}_{i\in N}, \mathsf{sk}_{\mathcal{S}_j}\big) \rightarrow (y_j, \pi_j, R_{\mathcal{S}_j}, \rho_j)$ *: given a set of public keys, shares and random values* $(\widehat{x_{ij}}, R_i, \mathsf{pk}_i)$ *for a list of users $\mathcal{U}_i$ in the subset $N \subseteq$*

$[1,n]$, *the aggregation algorithm performs all the key-agreements between* $\mathcal{U}_i$ *and* $\mathcal{S}_j$ *as* $\mathsf{KAgree}(\mathsf{sk}_{\mathcal{S}_j}, \mathsf{pk}_{\mathcal{U}_i}) \rightarrow \mathsf{s}_{ij}$*, the partial evaluation and proofs as:*

$$\begin{aligned} y_j &\leftarrow \mathsf{SS.Eval}\big(\{\widehat{x_{ij}}\}_{i\in N}\big) & \pi_j &= \sum_{i\in N} \mathsf{s}_{ij} \\ R_{\mathcal{S}_j} &= \prod_{i\in N} R_i & \rho_j &= \prod_{i\in N} R_i^{-\sum_{k\in N}^{k\neq i} \mathsf{s}_{kj}} \end{aligned} \tag{27}$$

The algorithm outputs $(y_j, \pi_j, R_{\mathcal{S}_j}, \rho_j)$.

- $\mathsf{Ver}\Big(t, \{\tau_{ij}\}_{i\in N}^{j\in M}, \big\{(y_j, \pi_j, R_{\mathcal{S}_j}, \rho_j)\big\}_{j\in M}\Big) \rightarrow \{y, \perp\}$ *: given the threshold* t*, a set of servers* M *with* $t{+}1 \leq |M| \leq m$*, given partial evaluations and proofs* $(y_j, \pi_j, R_{\mathcal{S}_j}, \rho_j)$ *and a set of verification coefficients* $\{\tau_{ij}\}_{i\in N}^{j\in M}$ *for a subset of users* N*, the verification algorithm verifies that for any* $\mathcal{S}_j, \mathcal{S}_j{}' \in M$*, it holds* $R_{\mathcal{S}_j} = R_{\mathcal{S}_j{}'} = R$*. If not,* Ver *outputs* $\perp$*. Otherwise, the algorithm verifies that for* ***all*** *the subsets* $T_i \subseteq M$ *of* $t{+}1$ *partial evaluations, the reconstruction algorithm* $\mathsf{SS.Recon}\big(t, \{y_j\}_{j\in T_i}\big)$ *returns always the same output* y*. If not,* Ver *outputs* $\perp$*. Otherwise, the algorithm computes*

$$\prod_{j\in M_l}^{i\in N} \tau_{ij} \stackrel{?}{=} \left(\prod_{j\in M_l} \mathsf{pk}_{\mathcal{S}_j}\right)^y \cdot \prod_{j\in M_l} R^{\pi_j} \cdot \rho_j \tag{28}$$

for all the $|M|$ *subsets* $M_l \subset M$ *such that* $|M_l| = |M| - 1$*. If any check fails, then the verification algorithm outputs* $\perp$*. Otherwise, the verification algorithm outputs* y*.*

Corollary 3. *NIVA allows the definition of the algorithm:*

- $\mathsf{OptVer}\Big(t, \{\tau_{ij}\}_{i\in N}^{j\in M}, \big\{(y_j, \pi_j, R_{\mathcal{S}_j}, \rho_j)\big\}_{j\in M}\Big) \rightarrow \{y, \perp\}$ *: given the threshold* t*, a set of servers* M *with* $t{+}1 \leq |M| \leq m$*, given partial evaluations and proofs* $(y_j, \pi_j, R_{\mathcal{S}_j}, \rho_j)$ *and a set of verification coefficients* $\{\tau_{ij}\}_{i\in N}^{j\in M}$ *for a subset of users* N*, the verification algorithm verifies that for any* $\mathcal{S}_j, \mathcal{S}_j{}' \in M$*, it holds* $R_{\mathcal{S}_j} = R_{\mathcal{S}_j{}'} = R$*. If not,* Ver *outputs* $\perp$*. Otherwise, the algorithm verifies that for* ***all*** *the subsets* $T_i \subseteq M$ *of* $t{+}1$ *partial evaluations, the reconstruction algorithm* $\mathsf{SS.Recon}\big(t, \{y_j\}_{j\in T_i}\big)$ *returns always the same output* y*. If not, the algorithm outputs* $\perp$*. Otherwise, the algorithm computes, for each* $\mathcal{S}_l \in M$*:*

$$\prod_{i\in N} \tau_{ij} \stackrel{?}{=} \big(\mathsf{pk}_{\mathcal{S}_l}\big)^y \cdot R^{\pi_l} \cdot \rho_l$$

If any check fails, then the algorithm outputs $\perp$*. Otherwise, it outputs* y*.*

Remark 13. *The main difference* w.r.t. Ver *is that* OptVer *takes the* $|M|$ *different subsets* M_l *to be defined as servers' singletons,* i.e. $M_l = \{\mathcal{S}_l\}$ *and* $|M_l| = 1$*. This reduces the amount of computation needed to verify Eq.* (28)*. The possibility of using* OptVer *might depend on* application constraints, e.g. *the server* Agg*'s outputs might not be directly published but further aggregated by a third party before reaching the final public verification.*

NIVA's is **correct** for both the verification algorithms Ver and OptVer.

NIVA Correctness. For any list of key pairs $\mathsf{Setup}(\lambda) \rightarrow (\mathsf{sk}_I, \mathsf{pk}_I)$ for any party I being a user $\mathcal{U}_i$ or server $\mathcal{S}_j$ for $i \in [1,n], j \in [1,m]$, for any user choice of secret inputs $x_i \in [0, p{-}1]$, for all computed shares

$$\mathsf{SGen}\big(x_i, \mathsf{sk}_{\mathcal{U}_i}, t, \{\mathsf{pk}_{\mathcal{S}_j}\}_{j=1}^m\big) \rightarrow \big(\mathsf{pk}_{\mathcal{U}_i}, \{\widehat{x_{ij}}\}_{j=1}^m, R_i, \{\tau_{ij}\}_{j=1}^m\big)$$

and for all aggregated values $(y_j, \pi_j, R_{\mathcal{S}_j}, \rho_j)$ computed as $\mathsf{Agg}\big(\{(\mathsf{pk}_{\mathcal{U}_i}, \widehat{x_{ij}}, R_i)\}_{i\in N}, \mathsf{sk}_{\mathcal{S}_j}\big)$, for any subset of users N, for any servers' subset $M \subseteq \mathsf{S}$ such that $|M| \geq t+1$, the verification algorithm, *i.e.* $\mathsf{Ver}\Big(t, \{\tau_{ij}\}_{i\in N}^{j\in M}, \big\{(y_j, \pi_j, R_{\mathcal{S}_j}, \rho_j)\big\}_{j\in M}\Big)$, finds that for any $\mathcal{S}_j, \mathcal{S}_j{}' \in M$, and for any subset T_i of $t+1$ partial evaluations, $\mathsf{SS.Recon}\big(t, \{y_j\}_{j\in T_i}\big)$ always returns the same y from the correctness of the secret sharing scheme. Finally, consider Eq. (26), the verification algorithm correctly verifies

$$
\begin{aligned}
\prod_{j\in M_l}^{i\in N} \tau_{ij} &= \prod_{j\in M_l}^{i\in N} \mathsf{pk}_{\mathcal{S}_j}^{x_i} R_i^{\mathsf{s}_{ij}} = \left(\prod_{j\in M_l} \mathsf{pk}_{\mathcal{S}_j}\right)^{\sum_{i\in N} x_i} \prod_{i\in N} R_i^{\sum_{j\in M_l} \mathsf{s}_{ij}} \\
&= \left(\prod_{j\in M_l} \mathsf{pk}_{\mathcal{S}_j}\right)^{y} \prod_{i\in N} R_i^{\sum_{k=i}^{j\in M_l} \mathsf{s}_{kj} + \sum_{k\in N, k\neq i}^{j\in M_l} \mathsf{s}_{kj} - \sum_{k\in N, k\neq i}^{j\in M_l} \mathsf{s}_{kj}} \\
&= \left(\prod_{j\in M_l} \mathsf{pk}_{\mathcal{S}_j}\right)^{y} \cdot \prod_{i\in N} R_i^{\sum_{i\in N}^{j\in M_l} \mathsf{s}_{ij}} \left(\prod_{i\in N} R_i\right)^{-\sum_{k\in N, k\neq i}^{j\in M_l} \mathsf{s}_{kj}} \\
&= \left(\prod_{j\in M_l} \mathsf{pk}_{\mathcal{S}_j}\right)^{y} \cdot \prod_{j\in M_l} \left(\prod_{i\in N} R_i\right)^{\sum_{i\in N} \mathsf{s}_{ij}} \prod_{j\in M_l} \rho_j \\
&= \left(\prod_{j\in M_l} \mathsf{pk}_{\mathcal{S}_j}\right)^{y} \cdot \prod_{j\in M_l} R^{\pi_j} \rho_j
\end{aligned}
\tag{29}
$$

for each subset $M_l \subset M$ with $|M_l| = |M| - 1$. The verification algorithm outputs y, thus proving the correctness of the scheme.

Trivially, the same is true whenever considering the subsets $M_l = \{\mathcal{S}_l\}$, *i.e.* the verification executed by the OptVer algorithm.

□

Theorem 9 (NIVA Security)**.** *If we assume the negligible probability $\epsilon_{\mathsf{dLog}_I}$ of solving the dLog_I problem for the input subset I and the additive homomorphic secret sharing scheme's security, then NIVA is **secure** (Def. 37).*

NIVA's Security - Thm. 9. Let dLog_I be a hard problem and assume the existence of an adversary $\mathcal{A}$ able to break the $\mathsf{Exp}^{\mathsf{sec}}_{\mathsf{NIVA}}(\mathcal{A})$ experiment of Def. 37.

The reduction $\mathcal{R}$ receives a dLog_I challenge $Z = g^z$, creates the m servers' key-pairs $(\mathsf{sk}_{\mathcal{S}_j}, \mathsf{pk}_{\mathcal{S}_j})$ and provides the corrupted to $\mathcal{A}$. $\mathcal{A}$ replies with the amount of challenged user n. The reduction therefore obtains the n users' key-pairs $(\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i})$ and samples $n-1$ secret inputs $\{x_i\}_{i=2}^n$ such that $\sum_{i=2}^n x_i = 0$.

Of these secrets, the reduction correctly computes the shares, for $i \in [2, n]$, as $\big(\mathsf{pk}_{\mathcal{U}_i}, \{\widehat{x_{ij}}\}_{j=1}^t, R_i\big) \leftarrow \mathsf{SGen}\big(x_i, \mathsf{sk}_{\mathcal{U}_i}, t, \{\mathsf{pk}_{\mathcal{S}_j}\}_{j=1}^m\big)$.

Regarding $i = 1$, $\mathcal{R}$ samples m uniformly random values $\widehat{x_{1j}}$ to be proposed as shares, correctly randomly sample r_1 and computes $R_1 = \mathsf{g}^{r_1}$. $\mathcal{R}$ computes the shares secrets s_{1j} and verification values τ_{1j} computed as:

$$
\tau_{1j} = Z^{\mathsf{sk}_{\mathcal{S}_j}} \cdot R_1^{\mathsf{s}_{1j}} \qquad \forall\, j \in [1, m] \tag{30}
$$

Observe that the computed τ_{1j} are equal to the correct verification coefficient obtained by using $x_1 = z$ as the secret input, formally:

$$
\tau_{1j} = Z^{\mathsf{sk}_{\mathcal{S}_j}} \cdot R_1^{\mathsf{s}_{1j}} = \mathsf{pk}_{\mathcal{S}_j}^{z} \cdot R_i^{\mathsf{s}_{1j}} \qquad \forall\, j \in [1, m] \tag{31}
$$

The reduction returns to $\mathcal{A}$ the shares $\left(\mathsf{pk}_{\mathcal{U}_i}, \{\widehat{x_{ij}}\}_{j=1}^{t}, R_i\right)$ and all the verification coefficients $\{\tau_{ij}\}_{j=1}^{m}$ for each $i \in [1, n]$.

Observe that the adversary $\mathcal{A}$ is unable to reconstruct the final aggregated value $y = \sum_{i=1}^{n} x_i$ since it only posses t shares out of the necessary $t+1$ required by the security of the secret sharing scheme. In other words, the t randomly generated shares of $\mathcal{U}_1$ cannot be used to (even) identify that they are not correctly computed since the provided communication and a correct execution have the same distribution. At this point, $\mathcal{A}$ replies with the guess $y^\star$ which is forwarded by $\mathcal{R}$ to the dLog_I challenger and observe that:

$$y^\star = \sum_{i=1}^{n} x_i = x_1 + \sum_{i=2}^{n} x_i = z + 0 = z$$

If $\mathcal{A}$ has a non-negligible advantage to win the $\mathsf{Exp}^{\mathsf{sec}}_{\mathsf{NIVA}}(\mathcal{A})$ experiment, $\mathcal{R}$ has the same non-negligible advantage to break dLog_I which is assumed to be hard, which is a contradiction. Thus:

$$\Pr\left[\mathsf{Exp}^{\mathsf{sec}}_{\mathsf{NIVA}}(\mathcal{A}) = 1\right] = \epsilon_{\mathsf{dLog}_I} < \mathsf{negl}$$

which proves that NIVA is secure.

□

Theorem 10 (NIVA Verifiability)**.** *Consider n users and m servers, with threshold t such that the order p of the cyclic group $\mathbb{G}$ used for the key-agreement does not divide $m-1$. Let $\mathcal{A}$ be a PPT adversary that maliciously controls $k < m$ servers, w.l.o.g. $\{\mathcal{S}_j\}_{j=1}^{k}$. It holds that NIVA is **verifiable** (Def. 38).*

NIVA's Verifiability - Thm. 10. Similarly to before, the adversary $\mathcal{A}$ provides $N^\star$, it is unable to reconstruct the final aggregated value $y = \sum_{i=1}^{n} x_i$ since it only posses t shares out of the necessary $t+1$ required by the security of the secret sharing scheme. Additionally, the choice of $N^\star$ does not have any security impact and it is only necessary for the challenger $\mathcal{C}$ to correctly compute the partial evaluations.

Consider $\{y_j, \pi_j, R_{\mathcal{S}_j}, \rho_j\}_{j=1}^{k}$ the honestly computed partial evaluations and observe that $\mathcal{A}$'s tamper $\{{y_j}^\star, {\pi_j}^\star, {R_{\mathcal{S}_j}}^\star, {\rho_j}^\star\}_{j=1}^{k}$ must, when reconstructed, obtain the final output $y^\star$ to be $y + \Delta$ for any subset of $t+1$ partial evaluation in $M^\star$ for some $\Delta \neq 0$. We can denote the tampers as, for any j, ${\pi_j}^\star = \pi_j + \epsilon_j$ and ${\rho_j}^\star = \rho_j \cdot \xi_j$. Observe that, all the ${R_{\mathcal{S}_j}}^\star$ must be equal to the correct R obtained from the mandatory uncorrupted server $\mathcal{S} \in M^\star$.

Let us focus on Eq. (28), and observe that for the subset M_l,

$$\begin{aligned}\prod_{j \in M_l}^{i \in N} \tau_{ij} &= \left(\prod_{j \in M_l} \mathsf{pk}_{\mathcal{S}_j}\right)^{y^\star} \cdot \prod_{j \in M_l} {R^{\pi_j}}^\star {\rho_j}^\star \\ &= \left(\prod_{j \in M_l} \mathsf{pk}_{\mathcal{S}_j}\right)^{y} \cdot \prod_{j \in M_l} R^{\pi_j} \rho_j \cdot \left(\prod_{j \in M_l} \mathsf{pk}_{\mathcal{S}_j}\right)^{\Delta} \cdot \prod_{j \in M_l} R^{\epsilon_j} {\xi_j}^\star\end{aligned}$$

where, in order to be correctly verified, it must hold, for each M_l,

$$\prod_{j \in M_l} \mathsf{pk}_{\mathcal{S}_j}^{\Delta} \cdot R^{\epsilon_j} {\xi_j}^\star = 1 \qquad \forall\, M_l \subset M \tag{32}$$

where each M_l can be split into the corrupted and the honest servers subsets, *i.e.* ${M_l}^\star = M_l \cap \{\mathcal{S}_j\}_{j \in [1,k]}$ and $\widetilde{M_l} = M_l \cap \{\mathcal{S}_j\}_{j \in [k+1,m]}$ where $\mu_l = |{M_l}^\star|$. We can therefore

expand Eq. (32) and obtain the system of equations:

$$\begin{cases} \prod_{j \in M_1{}^\star} \mathsf{pk}_{\mathcal{S}_j}^{\Delta} \cdot R^{\epsilon_j} {\xi_j}^\star = \prod_{j \in \widetilde{M_1}} \mathsf{pk}_{\mathcal{S}_j}^{-\Delta} \\ \vdots \\ \prod_{j \in M_\mu{}^\star} \mathsf{pk}_{\mathcal{S}_j}^{\Delta} \cdot R^{\epsilon_j} {\xi_j}^\star = \prod_{j \in \widetilde{M_\mu}} \mathsf{pk}_{\mathcal{S}_j}^{-\Delta} \end{cases} \tag{33}$$

which can be seen as $|M| = \mu$ equations in $|{M_l}^\star| \leq \mu - 1$ variables, *i.e.* $\mathcal{A}$ sees $\mathsf{pk}_{\mathcal{S}_j}^{\Delta} R^{\epsilon_j} {\xi_j}^\star$ as the variable g^{x_j} for each $\mathcal{S}_j \in {M_l}^\star$. This system has the same solution space as the system obtained by considering the exponents. In other words,

$$\begin{cases} \prod_{j \in M_1{}^\star} \mathsf{g}^{x_j} = \prod_{j \in \widetilde{M_1}} \mathsf{g}^{-\Delta \mathsf{sk}_{\mathcal{S}_j}} \\ \vdots \\ \prod_{j \in M_\mu{}^\star} \mathsf{g}^{x_j} = \prod_{j \in \widetilde{M_\mu}} \mathsf{g}^{-\Delta \mathsf{sk}_{\mathcal{S}_j}} \end{cases} \iff \begin{cases} \sum_{j \in M_1{}^\star} x_j = \sum_{j \in \widetilde{M_1}} -\Delta \mathsf{sk}_{\mathcal{S}_j} \\ \vdots \\ \sum_{j \in M_\mu{}^\star} x_j = \sum_{j \in \widetilde{M_\mu}} -\Delta \mathsf{sk}_{\mathcal{S}_j} \end{cases} \tag{34}$$

Denote the vector of variables with $\mathbf{x} = (x_1, \dots, x_{\mu_l})$ and the known coefficient with $\mathbf{b} = (-\mathsf{sk}_{\mathcal{S}_j} \Delta)_{j=\mu_l}^{\mu}$.By properly ordering the servers, Eq. (34) form the non-homogeneous linear system

$$\begin{pmatrix} D_{\mu_l} \\ 1_{(\mu-\mu_l)}^{\mu_l} \end{pmatrix} \cdot \mathbf{x} = \begin{pmatrix} 1_{\mu_l}^{(\mu-\mu_l)} \\ D_{(\mu-\mu_l)} \end{pmatrix} \mathbf{b} \tag{35}$$

which can be seen as the homogeneous system

$$D_\mu \cdot \begin{pmatrix} \mathbf{x} \\ \mathbf{b} \end{pmatrix} = \left(\begin{array}{c|c} D_{\mu_l} & 1_{\mu_l}^{(\mu-\mu_l)} \\ 1_{(\mu-\mu_l)}^{\mu_l} & D_{(\mu-\mu_l)} \end{array}\right) \cdot \begin{pmatrix} \mathbf{x} \\ \mathbf{b} \end{pmatrix} = 0_\mu \tag{36}$$

where 1_r^c denotes a matrix of r rows and c columns with all entries equal to one, 0_μ is the zero vector of length μ, $\mathsf{id}k$ is the identity matrix and $D_k = 1_k^k - \mathsf{id}k$, *i.e.* a k-square matrix of ones and null diagonal. Observe that we can verify the existence of a solution for Eq. (35) by using the Rouché-Capelli's theorem of Thm. 8. To do so, let us first prove that D_k always has maximum rank.

Lemma 4. *Consider the matrices defined over a field of characteristic p prime. For any $k \in \mathbb{N}$, $k > 0$ and such that p does not divide $k-1$, D_k has maximum rank,* i.e. *D_k is invertible.*

Proof. By reducing the matrix via the Euclidean algorithm,

$$D_k \implies \begin{pmatrix} 1 & 1 & \dots & 1 & 0 \\ 1 & \dots & \iddots & 0 & 1 \\ \vdots & & \iddots & & \vdots \\ 1 & 0 & \iddots & \dots & 1 \\ 0 & 1 & \dots & 1 & 1 \end{pmatrix} \implies \begin{pmatrix} 1 & 1 & \dots & 1 & 0 \\ 0 & \dots & \iddots & -1 & 1 \\ \vdots & & \iddots & & \vdots \\ 0 & -1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 1 & 1 \end{pmatrix} \implies \begin{pmatrix} 1 & 1 & \dots & 1 & 0 \\ 0 & 1 & \dots & 0 & -1 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & 1 & -1 \\ 0 & 0 & \dots & 0 & (k-1) \end{pmatrix}$$

from which we extract that the determinant of D_k is $\det(D_k) = k - 1$. Since p does not divide $k-1$, we have that $\det(D_k) \neq 0$ thus D_k is invertible and of maximum rank. □

By applying the lemma, we conclude that the system of Eq. (36) has rank μ while the one in Eq. (35) has rank $\mu_l = |M_l{}^\star| \leq \mu - 1 < \mu$. Rouché-Capelli guarantees that **no solution exists** that satisfies the system. Thus, $\mathcal{A}$ is unable to provide a correct tamper, thus NIVA is verifiable and $\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{ver}}_{\mathsf{NIVA}}(\mathcal{A}) = 1\right] = 0$. □

Observe that in the definition of the verification algorithm Ver, the servers' subset M *always allows* the existence of $|M| = \mu$ *different* subsets $M_l \subset M$ with $|M_l| = \mu - 1$ obtained as $M_l = M \setminus \{\mathcal{S}_l\}$ for each $\mathcal{S}_l \in M$. We require M to have at least $t + 1$ elements in order to execute the SS reconstruction SS.Recon.

Corollary 4. *NIVA achieves verifiability even in the case of using OptVer as the verification algorithm.*

NIVA's OptVer Verifiability - Corollary 4. The OptVer's correspondent system of Eq. (35) is

$$\begin{pmatrix} \mathsf{id}\mu_l \\ 0^{\mu_l}_{(\mu-\mu_l)} \end{pmatrix} \cdot \mathbf{x} = \begin{pmatrix} 0^{\mu_l}_{(\mu-\mu_l)} \\ \mathsf{id}(\mu - \mu_l) \end{pmatrix} \mathbf{b}$$

where 0^c_r denotes a matrix of r rows and c columns with all entries equal to zero. This can be seen as the homogeneous system

$$\mathsf{id}\mu \cdot \begin{pmatrix} \mathbf{x} \\ \mathbf{b} \end{pmatrix} = \left(\begin{array}{c|c} \mathsf{id}\mu_l & 0^{\mu_l}_{(\mu-\mu_l)} \\ 0^{\mu_l}_{(\mu-\mu_l)} & \mathsf{id}(\mu - \mu_l) \end{array}\right) \cdot \begin{pmatrix} \mathbf{x} \\ \mathbf{b} \end{pmatrix} = 0_\mu$$

Since the identity function has maximal rank, the two systems have different ranks and, exactly as in the proof, Rouché-Capelli guarantees us that no solution exists thus $\mathcal{A}$ is unable to provide a correct tamper. □

3.2 Additional Properties and Extensions

In this subsection, we discuss how additional properties presented by concurrent primitives/protocols [TM20, TLM18, BIK+17, XLL+20] apply to NIVA.

Multiple Executions. In the FL setting, it is required to execute the aggregation multiple times. NIVA is described for a single execution but the same generated key pairs allow the execution of multiple aggregation/verification calls.

Decentralization. Several published protocols [BIK+17, XLL+20] do not consider this decentralized scenario making their server a single point-of-failure, *i.e.* if the centralized server halts, the protocol cannot be terminated. NIVA decentralizes the aggregation between several servers and only a predefined amount is necessary for the correct reconstruction and verification of the output. This allows to overcome realistic problems such as *"complete the aggregation in case of failing servers"* or introduce *"responsibilities distribution"*, *i.e.* the servers might be owned by different *independent* entities and not by a single organisation.

Non-Interactivity and User Drop-Out. The aggregation problem discussed in this section can be solved either with an *interactive* protocol or a *non-interactive* primitive. The first allows the use of a *"challenge-response"* interaction that facilitates the computation of more complex verification protocols **but** introduces the **users' drop-out** problem, *i.e.* the user might drop-out during the communication thus are not able to finish the aggregation protocol, forcing the servers to abort the protocol. To overcome this issue, the protocol must be able to identify the drop-outs and recover the user's information to complete the aggregation or, if not possible, having a procedure for removing the user's initial participation. In a non-interactive solution, such as NIVA, a user cannot drop-out since there is no interaction. A dropping user in the non-interactive communication

is equivalent to a user that never participated. Thus any non-interactive solution is trivially able to overcome the users' drop-out problem.

On the other hand, interactivity allows to easily introduce *input's range proof* [PB10, CLZ12, LYAX18], *i.e.* a proof, generally in zero-knowledge, that allows the server to verify that the values obtained are indeed related to the user's secret input without revealing it. It might be possible to transform these zero-knowledge protocols into non-interactive proofs at the cost of introducing additional assumptions, *e.g.* the random oracle model for the Fiat-Shamir's transformation [FS87]. NIVA design's principle is simplicity with a small amount of assumption required; thus, allowing a more general deployment for different application/security models.

Authentication and Publishing. In this work, we do not consider *malicious* adversaries that are able to diverge from the correct communication. Similarly to the non-interactivity discussion, it might be possible to prevent active attacks by achieving communication authentication by, for example, force the registration of the servers' public keys on a public key infrastructure and using authenticated communications, *e.g.* communicating over a TLS channel. Additionally, NIVA requires the existence of an untamperable public *"space"* (*e.g.* a bulletin board) in which the partial proofs τ_{ij} to be used in the verification phase, will be stored. These requirements must be carefully considered whenever NIVA is used in a framework where active adversaries are a possibility.

Differential Privacy. Specific applications related to privacy preserving aggregation require a higher-level of privacy, especially when multiple aggregation outputs are published and from which it might be possible to infer information on a specific user/group. This is the case study for *differential privacy* [Dwo06] and the framework that implements it. Without entering tedious details, it is possible to utilize NIVA for differential private and distributed aggregation since it is possible to *introduce* the correctly sampled noise by using the additive-homomorphic property. The specific protocol for fairly and publicly generating the noise are tangent to NIVA's definition and to other abstract frameworks.

4 Implementation and Comparisons

In this section, we provide relevant statistics and performance measurements retrieved after implementing our primitive NIVA. We conclude by comparing NIVA with the results obtained by Segal *et al.*'s protocol [BIK+17] and Xu *et al.*'s VerifyNet [XLL+20]. NIVA is implemented as a prototype in Python 3.8.3 and we execute the tests on MacOS 10.13.6 over a MacBookPro (mid 2017) with processor Intel i5-7267U CPU @ 3.1GHz, with 16GB LPDDR3 2133MHz RAM, 256kB L2 cache and 4MB L3 cache. The source code of our implementation is publicly released[13]. For our experiments, the key agreement used is Diffie-Hellman over the elliptic curve secp256k1 and the additive homomorphic SS is Shamir's SS. The execution time is expressed in milliseconds (ms) and the bandwidth in kilobytes (kB).

The NIVA primitive is executed with respect to n users, m servers with the threshold parameter t and μ denoting the size of the verification set M. The total communication cost, *i.e.* users and servers' output data, is expected to be linearly dependent *w.r.t.* the numbers m and n, since each server has a constant size output, while the users are in total communicating nm shares x_{ij} **and** verification values τ_{ij}. Fig. 49 reports the expected behaviour.

Consider the metrics for a single user $\mathcal{U}$ and a server $\mathcal{S}$, depicted in Figures 50a and 50b. As expected, $\mathcal{U}$'s output data depends linearly on the amount of servers m.

[13] https://bitbucket.org/CharlieTrip/nivacode/src/main/

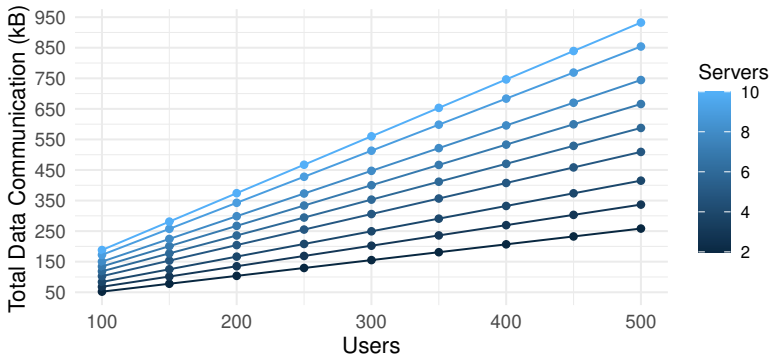
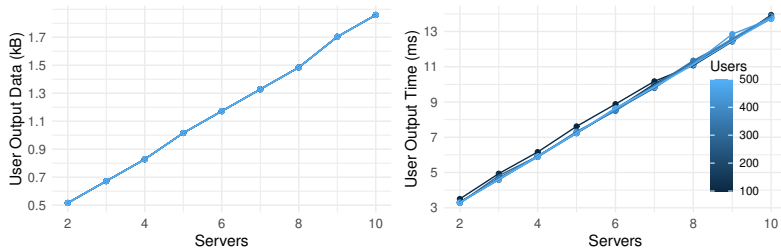
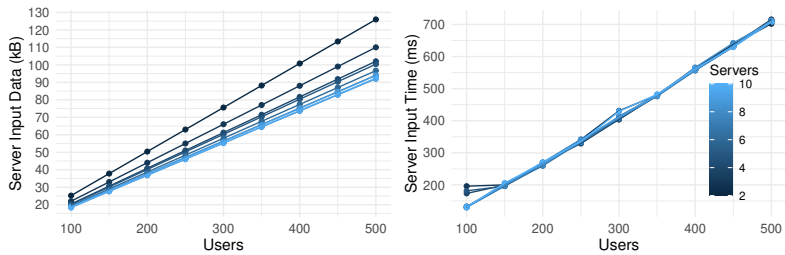


Figure 49: NIVA's total communication bandwidth for a different number n of users and m of servers and fixed $t = 1$ and $\mu = 2$.

The same applies for $\mathcal{S}$'s bandwidth and execution time, since they are linear *w.r.t.* the n users. Despite expecting $\mathcal{S}$'s input data to be always constant when considering different amount of servers and fixed n, our experiments present a decreasing $\mathcal{S}$'s data when increasing the amount of server. This is due to the approximation introduced by the Python data-measuring package used.



(a) User's data and computation time for a different number m of servers.

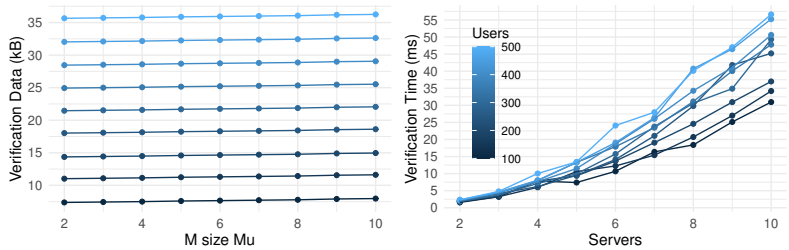


(b) Server's input data and timing per server for a different number n of users'.

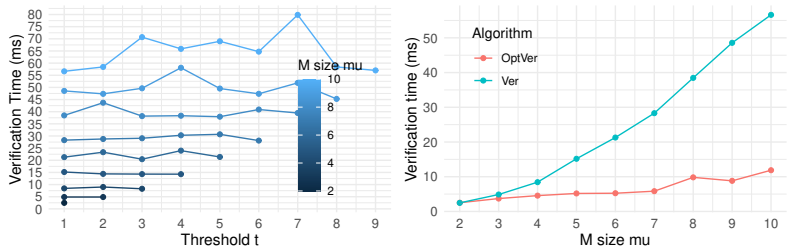
Figure 50: User and server's bandwidth and computation time performance.

As represented in Fig. 51b, the verification algorithm Ver has input data size proportional to the number μ of servers used in the verification. By considering the maximum verification set possible, Ver's execution time increases quadratically in the number of users and servers. In Fig. 51a, we observe that the optimal choice for μ is always $\mu = t + 1$. This is true because, for every $\mu \in [t + 1, m]$, a successful verification re-

quires *(1)* μ checks of the form of Eq. (28); and *(2)* $\binom{\mu}{t+1}$ calls to SS.Recon. The first is proportional *w.r.t.* the parameters n and μ, but it does not depend on t, while the latter has a maximal number of calls whenever μ is near the integer $2(t+1)$. This consideration suggests that it is optimal to minimise the verification set size μ to be $\mu = t+1$. Additionally, the optimized verification algorithm OptVer of Corollary 3 is always faster than Ver, due to the reduced amount of multiplications required during the verification of Eq. (28).



(a) Ver input data size and computation time for a different number n of users, verification set's size μ and amount of servers m.



(b) Ver's computation time for different μ and t and comparison between Ver and OptVer's computation time for different μ ($t = 1$).

Figure 51: Communication cost and execution time for Ver and OptVer.

4.1 Comparison to Related Work

We compare the performance of our solution with Segal *et al.*'s PPML [BIK+17] and Xu *et al.*'s VerifyNet [XLL+20] protocols. Segal *et al.*'s results are obtained from a Java implementation running on a Intel Xeon E5-1650 v3 CPU @ 3.50GHz, with 32 GB of RAM while, the server-side computations, Xu *et al.*'s are obtained from an Intel Xeon E5-2620 CPU @ 2.10GHz, 16GB RAM on the Ubuntu 18.04 operating system. Both of them have not publicly released their implementations, thus, making it hard to fairly compare the computation times. Additionally, since the considered related works are designed as interactive protocols, we can only compare total bandwidth/execution time and we will mainly focus on the user's and verification algorithm performance metrics since, in the FL scenario, the server enjoys high computational power.

In both the PPML and VerifyNet experiments, the users provide secret *vectors* of length K as input to the aggregation protocol and, additionally, the entries of the vector might be of small size, *e.g.* our implementation represents an integer with $B = 36$ bytes, while the *vector entries* considered in the PPML protocol are $b = 3$ bytes long. To fairly compare, we repeatedly execute NIVA $K\frac{b}{B}$ times in order to achieve the same amount of aggregated value bytes. In other words, we simulate the packing of

a vector of small integers into a single bigger integer, as described in the VerifyNet's implementation [XLL+20]. PPML assumes that the vector entries are of length $b = 3$ bytes, while VerifyNet was tested on entries of the same size B as NIVA. Since NIVA is the **only** decentralized primitive compared, we test it at the minimal distributed setting possible, *i.e.* $m = 2$ servers both needed for the reconstruction, or threshold $t = 1$.

VerifyNet uses as standard vector size $K = 1000$. Fig. 53a depicts that NIVA is more space efficient than VerifyNet whenever introducing a larger amount of users. Furthermore, NIVA requires a lower amount of users' data than VerifyNet. We should note though that whenever increasing the vector size K, it must be observed that NIVA has a slightly steeper angle, which means that there exists a vector size $\grave{k}$ from which VerifyNet becomes more efficient than NIVA. Differently, Fig. 53c collects the required user execution (computation) time in which NIVA results to be always more efficienct than VerifyNet.

PPML is defined with a standard vector of size $K = 10^5$, 100 times bigger than VerifyNet, and **does not** achieve the verification of the aggregated output. Additionally, each vector entry is described with $b = 3$ bytes, 12 times smaller than NIVA's input. As shown in Fig. 53b and Fig. 53d, our primitive seems to never be able to compete with the PPML protocol because of the elevated value K. PPML's protocol minimizes the communication cost, thus the execution time, for bigger vector sizes K, while it is linearly dependent on the number of users. In contrast, NIVA has a fixed user's communication cost that only depends on the vector size K and the amount of servers m. For this reason, we consider $K = 10^5$ and extrapolate the PPML's linear dependency between data and users n. We observe that NIVA overtakes PPML regarding both the user's execution time and the communicated data whenever the user size is $\sim 10^4$.

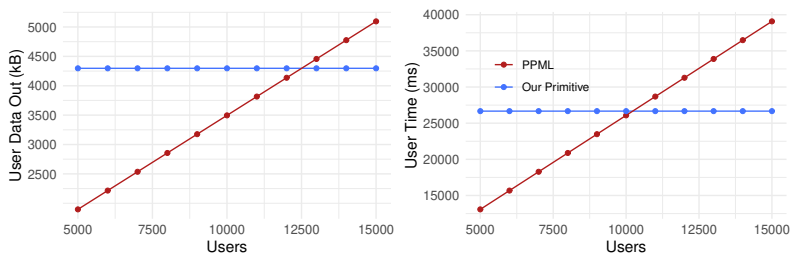
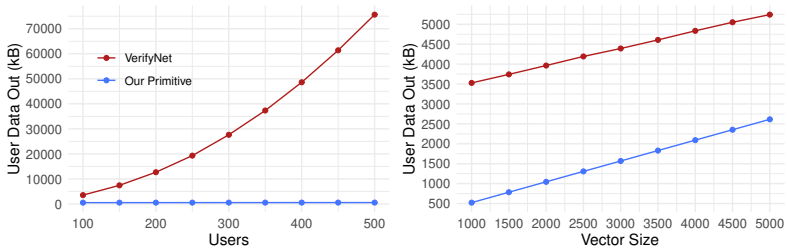
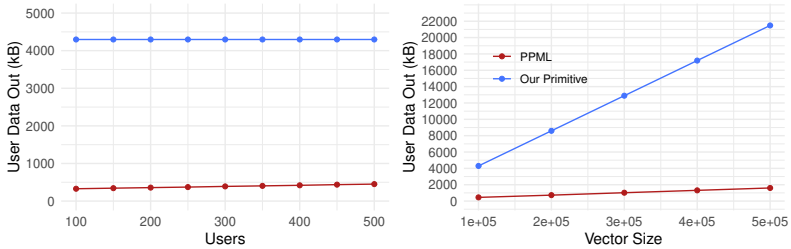


Figure 52: Extrapolated user's data usage and execution time for PPML and NIVA with fixed vector size $K = 10^5$.

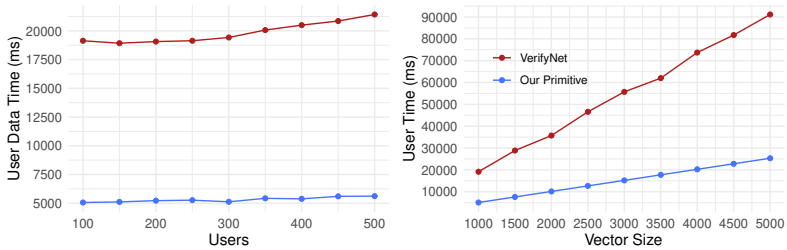
This allows us to conclude that NIVA is better suited than both PPML and VerifyNet for scenarios where the number of users n that participate in a FL model aggregation/update is substantial, *i.e.* over 10^5. For example, we have simulated a scenario where $n = 10^5$ users participate with a limited vector of $K = 1000$ entries of $b = 3$ bytes each and found out that NIVA has a constant user communication cost of $\sim$43.33kB and execution time of $\sim$282.5ms. In comparison and with the same hypothesis used for Fig. 52, PPML would require *each* user to communicate $\sim$31.55MB for a total of $\sim$4.33 minutes putting it over 3 order of magnitude worse than NIVA. Of course, NIVA's servers have a higher computational demand. In our experiments, each server took $\sim$106.56 hours to handle $\sim$4.00GB of data and the verification algorithm required $\sim$573.33MB of data from users and servers and was executed in $\sim$25.33 s. The reason for this high cost is the necessity to re-execute the primitive $K \cdot \frac{b}{B}$ times. This can be overcome by, for example, increasing B, thus, considering a key agreement based on *very-big* cyclic groups $\mathbb{G}$, such as an elliptic curve over a finite field of 512 bits which should allow to



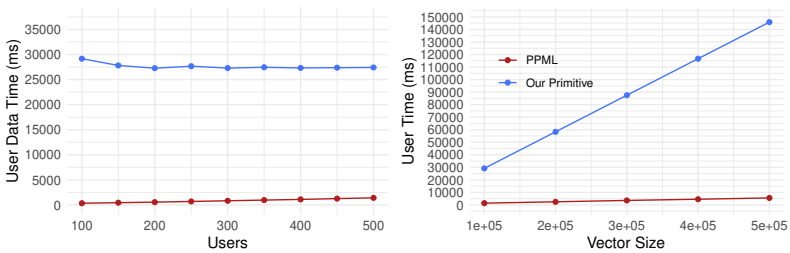
(a) User's data cost comparison between VerifyNet and NIVA for fixed vector size $K = 1000$ and number of users $n = 100$.



(b) User's data cost comparison between PPML and NIVA for fixed vector size $K = 10^5$ and number of users $n = 500$



(c) Timing comparison between VerifyNet and NIVA for fixed vector size $K = 1000$ and number of users $n = 100$.



(d) Timing comparison between PPML and NIVA for fixed vector size $K = 10^5$ and number of users $n = 500$.

Figure 53: Data and time comparisons between PPML, VerifyNet and NIVA.

almost double B from 36 to 64. It remains open if it is possible to extend NIVA to work more efficiently with vectors as secret inputs.

Turn Based Communication Channel

Carlo Brunetta[1], Mario Larangeira[2], Bei Liang[3], Aikaterini Mitrokotsa[1] and Keisuke Tanaka[2]

[1] Chalmers University of Technology, Gothenburg, Sweden
[2] Department of Mathematical and Computing Sciences, School of Computing, Tokyo Institute of Technology, Tokyo, Japan
[3] Beijing Institute of Mathematical Sciences and Applications, Beijing, China

Under Submission

Abstract: We introduce the concept of turn-based communication channel between two mutually distrustful parties with communication consistency, *i.e.* both parties have the same message history, and happens in sets of exchanged messages across a limited number of turns. Our construction leverages on *timed primitives*. Namely, we introduce a novel Δ-delay hash function definition in order to establish *turns* in the channel. Concretely, we introduce the one-way turn-based communication scheme and the two-way turn-based communication protocol and provide a concrete instantiation that achieves communication consistency.

Keywords: Time Puzzle, Delay, Hash Function, Consistency

1 Introduction

Communication channels are the core mediums allowing different parties to build dialogues. They can either be *physical* or *abstract*, *e.g.* electromagnetic wave propagation or a key exchange protocol that allows to *establish a secure communication channel*. Either the case, channels achieve different properties which can be related to the medium, *e.g.* reliability, energy efficiency, bandwidth, or based on the *"content"*, *e.g.* confidentiality, privacy or other. A fundamental and highly desirable property of a channel is *consistency*, *i.e.* different parties exchange messages which cannot be modified or repudiated in the future once the communication is over. In other words, whenever a message is shared, it is permanently fixed in the transcription. An example of a protocol that allows such a property is the *public bulletin board* which allows any party to publish any information on the *"board"*, while receiving a *"proof"* that guarantees the *integrity* that the information is indeed *published*. Recently, blockchains, or public ledgers [BGM16, KRDO17], have emerged as complex protocols that allow the instantiation of a public bulletin board, without relying on a central authority.

Their security relies on a specially purposed *consensus protocol*, which often requires assumptions of game-theoretic nature, *e.g.* the *proof-of-work* consensus protocol implies that an adversary does not have more than 51% of the available computing power at its disposal. Bulletin boards based on consensus protocols, albeit practical, suffer from significant delays when persisting entries. Notably, blockchain-based systems, typically suffer from scalability issues without a clear solution yet. Consequently, for time critical systems, blockchain-based bulletin boards may not be a useful alternative. An emerging technology, autonomous driving, illustrates the challenge between time-critical systems and blockchains. Autonomous driving in a real-world environment is a notoriously hard task because of the high number of variables that must be taken into account. Moreover, in such systems, communication between cars is a viable design approach. Different systems must communicate and coherently agree on their action plans.

Let us consider a simplified example where a car is overtaking another one. The one taking the action and surrounding cars must securely execute their algorithms while communicating to each other. All the communication between the cars should be timely available and guaranteed to be correct, *i.e.* could not be changed a posteriori, for audit purposes. The transcript of the whole communication could be used later, or even in court, for legal issues. A straightforward approach is to let vehicles be equipped with cryptographic primitives, such as digital signatures. Despite its feasibility, the aid of public key cryptography may not be an option in for some devices, in particular, resource restricted ones. Besides, it may require the use of Public Key Infrastructure (PKI) which may be, again, prohibitive for some systems.

One of the most basic building blocks in cryptographic literature are *hash functions*. They are used to guarantee data integrity and are widely employed in the computer science discipline in numerous applications. A natural question is whether such a building block would allow the construction of a *pair-wise communication channel*, avoiding the somewhat heavier cryptographic primitives earlier cited. An application relying only on hash functions could be significantly *"easier"*, since it would not be aided by public key cryptography schemes with PKI, typically more *"complex"* than their private key cryptography counterpart. Furthermore, it could also sidestep the early mentioned limitations of blockchain based protocols, yet providing a consistent and timely communication channel between two users. More succinctly, we investigate the following question:

is it possible to design a consistent channel between two parties ***without*** *using blockchain's assumptions* ***nor*** *public key infrastructure?*

Next, we detail the main approach of our idea which is to devise a *"turn"*, such that messages are exchanged only within the turns, and the proofs of submitted messages, similary to a bulletin board, are generated in order to guarantee consistency. The set of all turns of the channel, *i.e.* it contains a finite number of them are purposely related to each other. Therefore, they are not easily altered without affecting the overall transcript proofs of the exchanged communication.

Concept's Overview. All the communication is held over *time* which allows to *order* events during communication, *e.g.* message exchange. Commonly, our daily interaction is held over **continuous communication channels** in which the communicating parties can communicate at *any* point in time.

Our main idea, as depicted in Fig. 54, relies on providing a **turn-based communication channel (TBCC)** that forces the two parties to communicate in a *limited* amount of distinct *turns* separated by a Δ time interval. The interaction between the parties is slowed down by the necessity of *waiting for the next turn*, contrary to the almost-instantaneous reply ability of continuous channels.

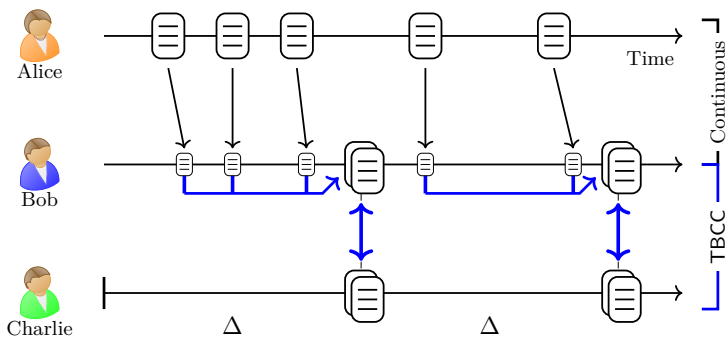


Figure 54: A continuous and TBCC channel, the messages are gathered in "blocks", and each block, and its set of messages, is confirmed only at the end of each turn.

To do so, we assume the existence of functions that *"computationally"* create time delays and are used to extend the hash function definition and introduce the Δ-delay hash function, which paves the way to the construction of **time-lock puzzles** in the spirit of Mahmoody *et al.* [MMV11], *i.e.* a primitive that allows Alice P_A to generate a puzzle-solution pair (y, π), send the puzzle y to Bob P_B that spends a time Δ to compute the solution π. Concretely, Δ is the turn interval in our TBCC construction. The novel feature provided by TBCC is that P_A knows the solution π in advance and can use it to *"commit"* to a message m. By releasing m and the puzzle y, P_B must invest Δ amount of time in computing π *before being able* to verify the validity of m. The early described *timed-commitment* is the stepping stone of our first construction for a **one-way turn-based scheme** that allows the communication of blocks of messages in turns in a single direction, *e.g.* from P_A to P_B. We show that if the one-way turn-based scheme is correct and tamper resistant, *i.e.* the adversary is unable to modify the past communication and/or the correctness of the exchanged messages, intuitively this yields to **communication consistency**, *i.e.* both parties have the same view of the exchanged messages even if the adversary delays/tampers any message. We define the **two-way TBCC protocol** as a *"two one-way scheme"* which allows a simpler extension of the properties to the protocol, *i.e.* correctness, tamper resistance, sequentiality and consistency. Additionally, we introduce the concept of **turn synchronisation**,

i.e. the two communicating parties must always agree in which *shared* turn they are communicating. The protocol can further provide a **recovery procedure** that allows the communicating parties to fix the last-turn messages in case of a communication error or an adversarial tamper. We summarise our ideas and contributions in Fig. 55.

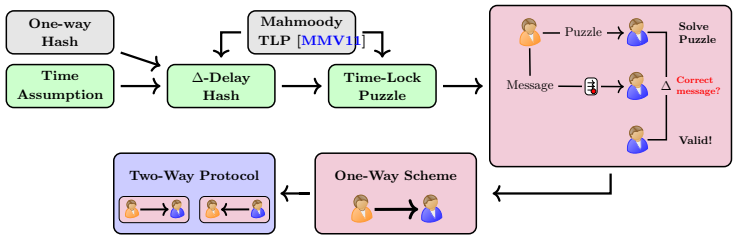


Figure 55: Roadmap of our contributions where we depict in gray the common assumption and definitions, in green our assumptions and basic primitives, in purple our main idea and construction and in blue our main contributions.

1.1 Related Work

Blockchains and Bulletin Boards. The blockchain data structure is commonly used in a distributed environment, where cryptographic primitives intersect with game theoretical assumptions in order to create a distributed database, where consistency comes for the orderly generation of blocks added to the structure. In the literature there are many examples of either *using* blockchains as a building block with new primitives, *e.g.* public verifiable proofs [SSV19], or applying existing cryptographic primitives into blockchains in order to achieve new functionalities [BBF19, KMS14]. Other focus is dedicated to the theoretical aspects related to the consensus mechanism or the blockchains' theoretical model [GKL15].

Time and Cryptographic Primitives. Cryptography and timing are long time distinct aspects that are commonly not considered together. Rivest *et al.* [RSW96] described the possibility of using time to create a *cryptographic time-capsule*, *i.e.* a ciphertext that will be possible to decrypt after a specified amount of time. Their work defines the concept of *time-lock puzzles*, where timing is achieved by cleverly tweaking the security parameters of some secure cryptographic primitives, *e.g.* choose a specific parameter λ such that the computational complexity of a specific problem is solvable by a real machine in reasonable time. Boneh *et al.* [BN00] presented the concept of timed commitments, *i.e.* a commitment scheme in which at any point, by investing an amount of effort, it is possible to correctly decommit into the original message. The main conceptual difference with respect to previous works is that, in this work, timing properties are achieved by forcing the algorithm to compute a naturally sequential mathematical problem. From a different perspective, Mahmoody *et al.* [MMV11] defined time-lock puzzles by just assuming the existence of timed primitives.

In the last years, many community efforts are spent into the definition of *verifiable delay functions* (VDFs), *i.e.* to compute a timed function and be able to verify the correct computation of it. There are multiple instantiations of this primitive in the literature, *e.g.* Lenstra *et al.*'s random zoo [LW15], a construction using randomized encoding by Bitansky *et al.* [BGJ+16] or Alwen-Tackmann's theoretical consideration regarding *moderately hard functions* [AT17]. The VDF's formal definition is given by Boneh *et al.* [BBBF18], subsequent papers provide additional properties for these time

related primitives such as Malavolta-Thyagarajan's homomorphic time-puzzles [MT19] or the *down-to-earth* VDF instantiation by Wesolowski [Wes19].

Timing Model. Perhaps the closest set of works to our study deals with the Timing Model as introduced by Dwork *et al.* [DNS04], and used by Kalai *et al.* [KLP07]. While they do present similarities to our work, *e.g.* the idea of *"individual clock"*, they also present significant differences. For instance, while in [DNS04, KLP07] every party in the real execution is equipped with a *"clock tape"*, extending the Interactive Turing Machine (ITM) with clocks, in our model the parties are regular ITMs, that perform computations in order to realize a *"single clock"* used by the ideal functionality. Additionally, our work also shares similarities with Azar *et al.* [AGP16] work on *ordered MPC*, which studies delays and ordered messages in the context of MPC. Our framework is positioned between both models as it focuses on turns equipped with a message validating mechanism, which is a different approach.

Recently, a concurrent and theoretical work by Baum *et al.* [BDD+20] formalizes the security of time-lock puzzles in the UC framework. More concretely they introduce the *UC with Relative Time* (RUC), which allows modelling relative delays in communication and sequential computation without requiring parties to keep track of a clock, in contrast to Katz *et al.*'s [KMTZ13] approach which models a *"central clock"* that all parties have access. The main contribution introduces a *semi-synchronous* message transmission functionality in which the adversary is aware of a delay Δ used to schedule the message exchanges, while the honest parties are not aware. In their work, composable time-puzzle realizes such novel functionality, and yields UC secure fair coin flips and two party computation achieving the notion of *output independent abort*. They focused on composable primitives and therefore have to rely on a constrained environment, *i.e.* it has to signal the adversary and activate every party at least once. Another theoretical difference is the focus of the order and turns but not in relative delays as in [BDD+20].

Baum *et al.* state as future work a possible extension to their transmission model in which all the parties have a *local clock* that would allow to always terminate any protocol. Our paper tackles that extension and provides a tangible instantiation of the extended model.

Paper Organisation. Sec. 2 states the preliminaries and time-complexity assumption. Sec. 3 defines the one-way and two-way TBCC protocol and related properties. Sec. 4 presents a collectively flip-coin protocol between two parties.

2 Preliminaries

In this section, we present notations and assumptions used throughout the paper.

We denote vectors with bold font, *e.g.* $\mathbf{v}$, and $\mathsf{Pr}[E]$ the probability of the event E. Let $\{0,1\}^*$ be the binary strings space of arbitrary length, $\mathbb{N}$ the natural numbers, $\mathbb{R}$ the real numbers and $\mathbb{R}_+$ the positive ones. Let $[a,b]$ denote intervals between a and b and $x \leftarrow_R X$ the random uniform sampling in the set X. Let $\mathsf{negl}(\lambda)$ denote a negligible function in λ, *i.e.* $\mathsf{negl}(\lambda) = O(\lambda^{-c})$ for every constant $c > 0$. We omit λ whenever obvious by the context.

Definition 40 (One-Way Hash Function [KL08])**.** *Let $n \in \mathbb{N}$. The function $H : \{0,1\}^* \to \{0,1\}^n$ is a **one-way hash function** if it satisfies the properties:*

- ***Preimage resistance:*** *for any $x \leftarrow_R \{0,1\}^*$ and $y := H(x)$, for any PPT adversary $\mathcal{A}$ that, on input y, outputs x', it holds that $\mathsf{Pr}[H(x') = y] < \mathsf{negl}$;*
- ***2nd Preimage resistance:*** *for any $x \leftarrow_R \{0,1\}^*$, $y := H(x)$, for any PPT adversary $\mathcal{A}$ that, on input x, outputs $x' \neq x$, it holds $\mathsf{Pr}[H(x') = y] < \mathsf{negl}$;*

Complexity and Time. Let time be modelled as the positive real numbers $\mathbb{R}_+$. At the core of our construction, we must assume the existence of a measure $\mu(\cdot)$ that plays the role of a *"bridge"* between *complexity and timing*. Formally,

Assumption 7. *Given a model of computation $\mathcal{M}$, there exists a measure $\mu(\cdot)$ that takes as input an $\mathcal{M}$-computable function f with input x and outputs the amount of time $\mu(\mathsf{f}, x) \in \mathbb{R}_+$ necessary to compute $\mathsf{f}(x)$ in the model $\mathcal{M}$. If $\mathsf{f}^\star(x)$ is a* probabilistic *function with input x and internal randomness r, then there exists $\mathsf{f}(x; r)$* deterministic *function that executes $\mathsf{f}^\star(x)$ with fixed randomness r.*

Informally, given a model of computation, *e.g.* Turing machines, quantum computers, *"pen-and-paper"*, it is possible to measure *"how much time does it take"* to compute $\mathsf{f}(x)$ both in the cases when f is deterministic or probabilistic[14].

Another required assumption is the existence of a function family $\mathcal{F}$ of which functions always output the results after the same amount of time. Formally,

Assumption 8. *Given a model of computation $\mathcal{M}$ and associated $\mu(\cdot)$, there exists a function family $\mathcal{F}$ such that for any function $\mathsf{f} \in \mathcal{F}$, for any inputs x, x', f is* input-independent *with computing time $\mu(\mathsf{f})$,* i.e. *$\mu(\mathsf{f}) = \mu(\mathsf{f}, x) = \mu(\mathsf{f}, x')$.*

Through the remaining of this work, we consider timing as the output of $\mu(\cdot)$ applied on input-independent functions. Whenever not specified, a *hard* problem is a problem of which solution, computed via f, has *large* computation time $\mu(\mathsf{f})$.

The *timed* one-way hash function extends the hash's properties of Def. 40.

Definition 41 (Δ-Delay One-Way Hash Function)**.** *Let $n \in \mathbb{N}$. The function $\overline{H} : \{0,1\}^* \to \{0,1\}^n$ is a* ***Δ-delay one-way hash function*** *if it is input-independent as described in Assumption 8 and, in addition to the properties of Def. 40, the following property also holds:*

- ***Δ-Delay:*** *for any PPT adversary $\mathcal{A}$ that takes an input x and outputs y which runs in time $\mu(\mathcal{A}, x) < \Delta = \mu(\overline{H})$, it holds that $Pr[y = \overline{H}(x)] < \mathsf{negl}$.*

Observe that, in order for the Δ-delay's property to make sense, the length of x might require to be limited, *e.g.* x must be polynomial. We omit such detail and always consider delay hash functions with the appropriate input space size.

Define the **time-lock puzzle** (TLP) as a *generate-solve* algorithm pair in which time plays a design/security aspect. Our definition is inspired by Azar *et al.* [AGP16] and, more specifically, we consider the construction presented by Mahmoody *et al.*'s [MMV11] in the random oracle (RO) model. The provided TLP generates $m+1$ *sequential* puzzles, *i.e.* a list of **partial puzzle** y_i of which **partial solution** π_i is necessary in order to solve the next partial puzzle y_{i+1}.

Definition 42 (Time-Lock Puzzle)**.** *Let $m \in \mathbb{N}$, security parameter λ and $\Delta \in \mathbb{R}_+$ be the desired time delay. Let $\overline{H} : \{0,1\}^* \to \{0,1\}^n$ be a Δ-delay hash function for some $n \in \mathbb{N}$. Let the algorithms $(\mathsf{GenPuz}, \mathsf{SolPuz})$ define a $(m\Delta)$* ***time-lock puzzle*** *$(m\Delta$-TLP$)$ as:*

- *$\mathsf{GenPuz}(\lambda, (m, \Delta)) \to (\mathbf{y}, \pi)$: the generation algorithm randomly samples $m+1$ bit-strings $x_i \in \{0,1\}^n$ and it computes the hash $\overline{H}(x_i)$ for $i \in [0, m]$. The algorithm outputs the list of partial puzzles and partial solutions:*

$$(\mathbf{y}, \pi) \coloneqq \Big(\big(x_0, \overline{H}(x_0) \oplus x_1, \dots, \overline{H}(x_{m-1}) \oplus x_m\big), (x_0, x_1, \dots, x_m) \Big);$$

[14] Observe that the *same* computational problem might have *different* timing, *e.g.* solving a classic-secure discrete logarithm instance is infeasible on a classical computer while it is theoretically feasible on a quantum computer.

- $\mathsf{SolPuz}(\mathbf{y}, k, (\pi_0, \dots, \pi_{k-1})) \rightarrow \pi_k$: *the algorithm parses* $\mathbf{y}$ *into* $(y_0, y_1, \cdots, y_m)$, $k \in [1, m]$ *and the known partial solutions* $(\pi_0, \dots, \pi_k)$. *It then outputs the partial solution* $\pi_k \coloneqq y_k \oplus \overline{H}(\pi_{k-1})$ *where* $\pi_0 \coloneqq y_0$.

The following three properties must hold:

- ***Correctness****: for every delay* Δ, *security parameter* λ *and* $m, n \in \mathbb{N}$, *for every puzzle* $(\mathbf{y}, \pi) \leftarrow \mathsf{GenPuz}(\lambda, (m, \Delta))$, *for every* $k \in [1, m]$, *it holds that*

$$Pr[\mathsf{SolPuz}(\mathbf{y}, k, (\pi_0, \dots, \pi_{k-1}))=\pi_k] = 1$$

- ***Timing****: for every delay* Δ, *security parameter* λ *and values* $m, n \in \mathbb{N}$, *for every puzzle* $(\mathbf{y}, \pi) \leftarrow \mathsf{GenPuz}(\lambda, (m, \Delta))$, *for every* $k \in [1, m]$ *it holds that* $\mu(\mathsf{SolPuz}) = \Delta$ *and generating the puzzle is faster than solving it,* i.e.

$$\mu(\mathsf{GenPuz}) \leq m \cdot \mu(\mathsf{SolPuz})$$

- ***Locking****: for every delay* Δ, *security parameter* λ *and values* $m, n \in \mathbb{N}$, *for every puzzle* $(\mathbf{y}, \pi) \leftarrow \mathsf{GenPuz}(\lambda, (m, \Delta))$, *for every* $k \in [1, m]$ *and adversary* $\mathcal{A}$ *that solves the* k*-th partial puzzle,* i.e. $\mathcal{A}(y, k, (\pi_0, \dots, \pi_{k-1})) = \pi_k$, *it holds that* $\mu(\mathcal{A}) < \Delta$ *with only negligible probability.*

The $(m\Delta)$-TLP describes a sequence of sequential puzzles that must be solved one at a time. The timing property guarantees that the SolPuz algorithm requires a specific Δ amount of time to be executed and that generating the whole puzzle takes less time than solving all the m puzzles. The locking property guarantees that any adversary $\mathcal{A}$ is unable to solve the partial puzzle in less time than Δ which implies, intuitively, that SolPuz is the *most optimised* algorithm for solving the partial puzzle y_i. If a better solving algorithm SolPuz' exists with solving time $\Delta' < \Delta$, then $(\mathsf{GenPuz}, \mathsf{SolPuz}')$ is a $(m\Delta')$-TLP while $(\mathsf{GenPuz}, \mathsf{SolPuz})$ cannot satisfy the locking property.

3 Instantiating the Turn Based Communication Channel

In this section, we discuss the core concepts of **timed disclosure**, **turns block** and **communication consistency**, later used to fully instantiate one and two-way TBCC, from a time-lock puzzle based on a Δ-delay hash function.

Timed Disclosure and Message Block. Consider a Δ-delay hash function and the related time-lock puzzle (y, π) as defined in Def. 42. Alice generates and publishes the puzzle y. On receiving y, Bob starts solving it. Within the amount of time Δ, only Alice knows the solution π, which allows her to produce an efficient digest $\xi = \mathsf{H}(\mathsf{m}, \pi)$ for any message m that she wants to communicate with Bob. At this stage, Bob is unable to compute the same digest because he does not know π. The *"timed disclosure"* is achieved whenever Bob finds the solution π which enables him to accept or reject the previously received message by verifying the correctness of the digest ξ. *Timing is key* for the security of the disclosure: Alice must use the knowledge **before** it is disclosed and, on the other hand, Bob should reject anything that uses such secret **after** the disclosure. Differently, **only** after Δ time, Bob can check which are the correct messages that are blinded to the specific solution π and can collect them into a **turn block**. Whenever we consider that Alice can publish a sequential time-lock puzzle in which one partial solution π_i is the *starting point* for the next partial puzzle y_{i+1}, Bob must filter and accept the received messages into a block every Δ amount of time therefore creating the concept of **turns** and relative message blocks. This **turn point-of-view** is possible because of the *sequential timed disclosure* that can be seen as a *"clock that ticks"* every Δ amount of time. This means that the communication is

one-way, from Alice to Bob. Alice does *not see* the turn because all the partial solutions are known to her and therefore she is able to generate any possible message-digest pair at any time, see Fig. 56.

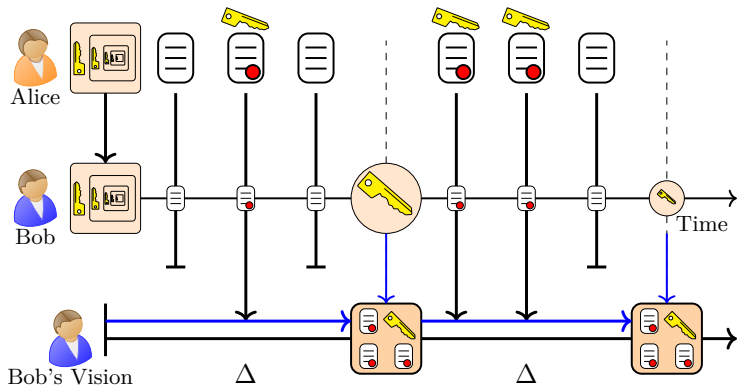


Figure 56: One-way channel scheme representation. Alice shares a time-lock puzzle with Bob and then sends messages of which some are correctly binded with the next puzzle's partial solution. With that solution, Bob is able to filter out the correct messages. Since this is done every Δ time, in Bob's eyes is as if he is receiving messages in turns.

Block of Messages and Communication Consistency. The next step is to create a *two-way* communication between Alice and Bob by allowing them to instantiate two independent one-way TBCC channels between each other, *i.e.* by exchanging time-lock puzzles and communicating message-digest pairs that are accepted and personally saved in blocks. These blocks are not stored in a trusted third party service but Alice and Bob have their own local copy of the exchanged message history and this means that it is required to provide a procedure to guarantee **consistency** between the copies. Consider our communicating Alice and Bob to be in the i-th turn, *i.e.* at the end of the turn they will create the i-th block. Naively, to achieve consistency of all blocks, every message, of the current block, should be bound to the *previous and future block*. For the *previous block*, they include a digest h_{i-1} of the previous block in every message they share in order to correctly verify that both have the same previous block vision. When the i-th turn ends, they separately create their own block-vision which could be different. When they enter the $(i+1)$-th turn, they will have to share the previous block digest h_i and they will see that the values are different. They will therefore start a **recovery phase** by publishing the content of the i-th block. At this point in time, the message's digest ξ_i can be tampered by anyone since the partial solution π_i is publicly known. For this reason, for every message we define a second digest σ_i that binds such message with the *next turn/future block* solution π_{i+1}. This procedure allows every party to understand *"who is cheating"* or *"where the errors are"*. In this way it is possible to abort the communication at any point in time, whenever a malicious party hijacks the channel. All the parties are thus forced to honestly participate if they want to maintain the channel up.

3.1 One-Way TBCC Instantiation

In this section, we instantiate the turn-based one-way channel from Alice to Bob. A *"channel"* is any collection of parameters that allows to participate into the communic-

ation, *e.g.* whenever a list of parameters is published, anyone can use them to correctly parse future messages shared using them.

Definition 43. *The* ***one-way channel scheme*** *is defined with the PPT algorithms* $(\mathsf{setup}, \mathsf{send}, \mathsf{ext}, \mathsf{turntoken}, \mathsf{valid\text{-}ver}, \mathsf{tamper\text{-}ver})$ *as:*

- $\mathsf{setup}(\lambda, \Delta, n) \rightarrow (\mathcal{C}, \mathcal{C}_{priv})$*: to setup the communication channel,* P_A *parses the security parameter* λ*, the delay* Δ *and the number of turns* n *The setup algorithm outputs the public and private channels* $(\mathcal{C}, \mathcal{C}_{priv})$*;*
- $\mathsf{send}(\mathcal{C}_{priv}, \mathsf{m}, v, \mathsf{t}) \rightarrow (\xi, \mathsf{aux})$*: the send-message algorithm takes in input the private channel information* $\mathcal{C}_{priv}$*, a message* m *with validity* $v \in \{0,1\}$ *and the turn* $\mathsf{t} < n$*. The algorithm outputs the message correctness proof* ξ *and the channel auxiliary information* aux*.*
- $\mathsf{turntoken}(\mathcal{C}, \mathsf{t}, \{x_0, \ldots, x_{\mathsf{t}-1}\}) \rightarrow x_{\mathsf{t}}$*: this algorithm is executed* at the beginning *of turn* t*. The algorithm parses the channel* $\mathcal{C}$*, the current turn* t *and the set of previously computed turn tokens* $\{x_0, \ldots, x_{\mathsf{t}-1}\}$*, after* Δ *amount of time, the algorithm outputs the turn token* x_{t}*.*
- $\mathsf{valid\text{-}ver}(\mathcal{C}, \mathsf{t}, \mathsf{m}, \xi, x_{\mathsf{t}}) \rightarrow \{0, 1\}$*:* at the end *of the* t*-th turn, the validity verification takes as input a message* m *and its proof* ξ *and the turn token* x_{t}*. The algorithm outputs the validity* v *for the sent message* m *with proof* ξ*;*
- $\mathsf{tamper\text{-}ver}(\mathcal{C}, \mathsf{t}, M_{\mathsf{t}-1}, \mathsf{m}, \mathsf{aux}, \xi) \rightarrow \{0, 1\}$*:* during *the* t*-th turn, the tamper verification algorithm takes in input the public channel* $\mathcal{C}$*, the current turn* t*, the* ordered block *of messages* $M_{\mathsf{t}-1}$ *which is the list of valid messages for the turn* $\mathsf{t}-1$*, a sent message* m *with proof* ξ *and auxiliary information* aux*. The algorithm verifies if the sent message* m *correctly relates to the previously sent messages contained in the block* $M_{\mathsf{t}-1}$*, thus outputting* 1 *when this is achieved, otherwise* 0*.*
- $\mathsf{ext}(\mathcal{C}, \mathcal{C}_{priv}, \mathsf{t}) \rightarrow x_{\mathsf{t}}$*: the extraction algorithm takes as input the public channel* $\mathcal{C}$*, the private channel* $\mathcal{C}_{priv}$ *and a turn* $\mathsf{t} \leq n$ *and outputs the turn token* x_{t}*, without investing any multiple of* Δ *time;*
- $\mathsf{backward\text{-}ver}(\mathcal{C}, \mathsf{t}, M_{\mathsf{t}-1}, l) \rightarrow \{0, 1\}$*: the recovery algorithm takes as input the public channel* $\mathcal{C}$*, the current turn* t*, the previous ordered block* $M_{\mathsf{t}-1}$ *of* $b_{\mathsf{t}-1} = |M_{\mathsf{t}-1}|$ *valid messages* m_i *and an index* $l \in [1, b_{\mathsf{t}-1}]$*. The algorithm outputs if the* l*-th message* $\mathsf{m}^\star$ *in the block* $M_{\mathsf{t}-1}$ *is a correct message for the block* $M_{\mathsf{t}-1}$ at the end *of turn* t*.*

Let us explain how the definition is used to generate a communication channel from Alice P_A to Bob P_B, as depicted in Fig. 57. First, P_A executes setup for an agreed delay Δ and amount of turns n, and obtains the channels $(\mathcal{C}, \mathcal{C}_{priv})$, *e.g.* the public channel $\mathcal{C}$ can consist of P_A's public key and public parameters while the private channel $\mathcal{C}_{priv}$ contains P_A's private key. The knowledge of $\mathcal{C}_{priv}$ allows P_A to quickly compute each turn token x_{t} *directly* as $\mathsf{ext}(\mathcal{C}, \mathcal{C}_{priv}, \mathsf{t})$ while P_B must *sequentially* compute them as $\mathsf{turntoken}(\mathcal{C}, \mathsf{t}, \{x_0, \ldots, x_{\mathsf{t}-1}\})$ and obtain them every Δ amount of time, similarly to a periodic scheduling process. Whenever P_A sends the message m in a turn t, she executes send for a *valid* message in the t *turn* and sends to P_B the tuple $(\mathsf{m}, \xi, \mathsf{aux})$. P_B can execute $\mathsf{valid\text{-}ver}(\mathcal{C}, \mathsf{t}, \mathsf{m}, \xi, x_{\mathsf{t}})$ and verify the message validity only whenever P_B obtains the turn token x_{t}, computable only after $\mathsf{t} \cdot \Delta$ amount of time. This allows P_A to communicate several messages of which P_B cannot immediately verify the validity of m but it has to wait for $\mathsf{turntoken}$ to output the specific turn token x_{t} thus creating the view of turns of the channel.

Message Validity. The sender's inputs are the validity value v, a bit which indicates if the message is considered valid or not, along with the message m itself and the choice of turn t. Only when the turn t ends, the receiver can verify the validity of the message via the $\mathsf{valid\text{-}ver}$ algorithm and the turn token x_{t}.

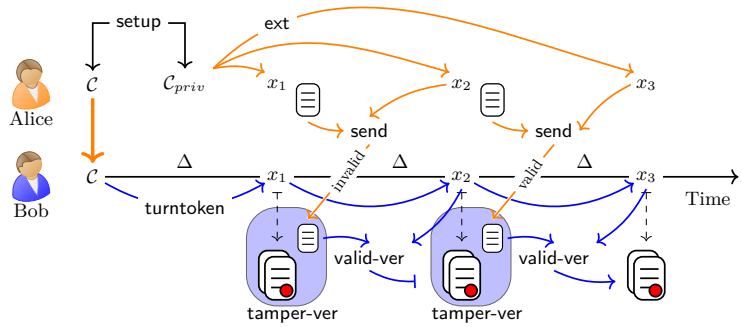


Figure 57: One-Way TBCC scheme usage: Alice submits the public channel $\mathcal{C}$ to Bob, and keeps the private information $\mathcal{C}_{priv}$. On each end of turn, Bob verifies the received messages in order to prevent the addition of invalid messages in the channel.

Definition 44 (Channel Correctness/Message Validity)**.** *Assume a turn* $\mathsf{t} \leq n$ *in a* n*-turn channel generated by the algorithms of Construction 1, then for all message/validity pairs* m *and* v*, the channel is said to be correct if*

$$Pr\left[\mathsf{valid\text{-}ver}(\mathcal{C}, \mathsf{t}, \mathsf{m}, \xi, x_{\mathsf{t}}) \neq v \;\middle|\; \begin{array}{l} \mathsf{setup}(\lambda, \Delta, n) \rightarrow (\mathcal{C}, \mathcal{C}_{priv}); \\ \mathsf{send}(\mathcal{C}_{priv}, \mathsf{m}, v, \mathsf{t}) \rightarrow (\xi, \mathsf{aux}); \\ \mathsf{ext}(\mathcal{C}, \mathcal{C}_{priv}, \mathsf{t}) \rightarrow x_{\mathsf{t}}; \end{array}\right] \leq \mathsf{negl}(\lambda)\,,$$

with probability computed over the random coins of setup*,* send*,* ext *and* valid-ver*.*

Sequentiality and Turn Definition. The turns of the channel rely on the time necessary to compute the token values x_{t} via turntoken, defined in the channel $\mathcal{C}$ during the general setup. Each computed turn-tokens x_{t}, allows the receiver to verify the validity and consistency of all received messages during the turn t, crucially, only at the end of the turn after the expected delay time Δ.

Definition 45 (Sequentiality)**.** *The channel is* Δ*-sequential if for any turn* t*, for any PPT adversary* $\mathcal{A}$ *running in time* $\mu(\mathcal{A}) < \Delta$*, the adversary wins the game* $\mathsf{Game}^{\mathcal{A},\Delta}_{seq}(\lambda, \mathsf{t}, n)$ *of Algorithm 3, with negligible advantage, namely,*

$$\left| Pr[\mathsf{Game}^{\mathcal{A},\Delta}_{seq}(\lambda, \mathsf{t}, n) = 1] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda)\,.$$

Algorithm 3 Sequentiality Game $\mathsf{Game}^{\mathcal{A},\Delta}_{seq}(\lambda, \mathsf{t}, n)$ for the adversary $\mathcal{A}$

1: Execute $\mathsf{setup}(\lambda, \Delta, n) \rightarrow (\mathcal{C}, \mathcal{C}_{priv})$;
2: Choose a random message m and validity $v \leftarrow \{0, 1\}$.
3: Execute $\mathsf{ext}(\mathcal{C}, \mathcal{C}_{priv}, i) \rightarrow x_i$ for $i \in [1, \mathsf{t}-1]$ and $\mathsf{send}(\mathcal{C}_{priv}, m, v, \mathsf{t}) \rightarrow (\xi, \mathsf{aux})$
4: $v^* \leftarrow \mathcal{A}\big(\mathcal{C}, \mathsf{t}, m, \xi, \mathsf{aux}, \{x_i\}_{i=1}^{\mathsf{t}-1}\big)$
5: Execute $\mathsf{ext}(\mathcal{C}, \mathcal{C}_{priv}, \mathsf{t}) \rightarrow x_{\mathsf{t}}$
6: If $\mathsf{valid\text{-}ver}(\mathcal{C}, \mathsf{t}, m, \xi, x_{\mathsf{t}}) = v^*$, output 1. Otherwise, 0

Last Turn Tamper Resistance. Given any $\mathsf{t} \leq n$ of a TBCC with public setup information $\mathcal{C}$, define the block $\mathsf{M}_{\mathsf{t}-1}$ as the set of all $j_{\mathsf{t}-1}$ messages in the turn $\mathsf{t}-1$

with respective auxiliary information $\mathsf{aux}_1, \dots, \mathsf{aux}_{j_{\mathsf{t}-1}}$ and sent proof $\xi_1, \dots, \xi_{j_{\mathsf{t}-1}}$. The algorithm $\mathsf{tamper\text{-}ver}(\mathcal{C}, \mathsf{t}, \mathsf{M}_{\mathsf{t}-1}, \mathsf{m}, \mathsf{aux}, \xi)$ checks, for any correctly computed message $(\mathsf{m}, \mathsf{aux}, \xi) \in \mathsf{M}_{\mathsf{t}}$, if it correctly relates to the previous turn block $\mathsf{M}_{\mathsf{t}-1}$ by spotting whenever this connection is tampered.

Definition 46 (Last Turn Tamper Resistance)**.** *During the turn $\mathsf{t} \leq n$ of a channel $\mathcal{C}$ between two honest parties with correct message blocks M_i for each turn $1 \leq i < \mathsf{t}$, $\mathcal{C}$ is tamper resistant, if for any PPT adversary $\mathcal{A}$, it holds*

$$Pr\left[\begin{array}{l} \mathit{tamper\text{-}ver}(\mathcal{C}, \mathsf{t}, M^{\star}_{\mathsf{t}-1}, m^{\star}, \mathit{aux}^{\star}, \xi^{\star}) = 1 | \\ \quad (M^{\star}_{\mathsf{t}-1}, m^{\star}, \mathit{aux}^{\star}, \xi^{\star}) \leftarrow \mathcal{A}(\mathcal{C}, \mathsf{t}, M_1, \dots, M_{\mathsf{t}-1}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

such that $M^{\star}_{\mathsf{t}-1} \neq M_{\mathsf{t}-1}$ and $\mathit{tamper\text{-}ver}(\mathcal{C}, \mathsf{t}, M_{\mathsf{t}-1}, m^{\star}, \mathit{aux}^{\star}, \xi^{\star}) = 1$. The probability is computed over the random coins of $\mathcal{A}$ and algorithm tamper-ver.

Remark 14. *Def. 46 is strictly dependent on the current turn being t, in the sense that it does not deal directly with tampering of messages in earlier turns than $\mathsf{t}-1$. By considering the definition for $1 \leq \mathsf{t} \leq n$, it covers all the turns for the channel,* ***except*** *the n-th turn creating the* ***last message tamper*** *that will not be possible to verify because there are* "no more turns", i.e. *the computation $\mathit{turntoken}(\mathcal{C}, n+1, \{x_i\}_{i=0}^{n})$ is not defined or the result must not be used.*

Communication Consistency. For any turn $\mathsf{t} \leq n$ of a one-way channel $\mathcal{C}$, the channel is *consistent until turn* $\mathsf{t}-1$ whenever the *valid* messages view between the parties is the same during the turn t, *i.e.* an adversary **must not** be able to force a wrong message history, regardless if it is the sender or the receiver.

Definition 47 (Consistency)**.** *During turn $\mathsf{t} \leq n$ of a one-way TBCC channel $\mathcal{C}$ between two parties with correct message blocks M_i for each turn $1 \leq i < \mathsf{t}$, the channel is* ***consistent until turn*** *$\mathsf{t}-1$, if for any PPT adversary $\mathcal{A}$, it holds*

$$\begin{array}{l} Pr[\mathit{tamper\text{-}ver}(\mathcal{C}, \mathsf{t}, M^{\star}_{\mathsf{t}-1}, m^{\star}, \mathit{aux}^{\star}, \xi^{\star}) = 1 | \\ \quad (M^{\star}_{\mathsf{t}-1}, m^{\star}, \mathit{aux}^{\star}, \xi^{\star}) \leftarrow \mathcal{A}(\mathcal{C}, \mathsf{t}, M_1, \dots, M_{\mathsf{t}-1}) \ ] \leq \mathsf{negl}(\lambda) \end{array}$$

such that $M^{\star}_{\mathsf{t}-1} \neq M_{\mathsf{t}-1}$, $\mathit{tamper\text{-}ver}(\mathcal{C}, \mathsf{t}, M_{\mathsf{t}-1}, m^{\star}, \mathit{aux}^{\star}, \xi^{\star}) = 1$ and for all the messages of the tampered block, along with auxiliary information and proof, i.e. $(m^{\star}_{j_i}, \mathit{aux}^{\star}_{j_i}, \xi^{\star}_{j_i}) \in M^{\star}_{\mathsf{t}-1}$, *it holds* $\mathsf{valid\text{-}ver}(\mathcal{C}, \mathsf{t}-1, m^{\star}_{j_i}, \xi^{\star}_{j_i}, x_{\mathsf{t}-1}) = 1$ *The probability is computed over the random coins of $\mathcal{A}$, tamper-ver and* valid-ver.

3.2 One-Way Channel Instantiation.

Let $\Delta \in \mathbb{R}_+$ be a time-delay and $n \in \mathbb{N}$ a maximal turn number, both chosen by Alice, denoted with P_A. Let H and $\overline{\mathsf{H}}$ be respectively regular and Δ-delay hash functions. Let $(\mathsf{GenPuz}, \mathsf{SolPuz})$ be the $(n\Delta)$-TLP of Def. 42 based on $\overline{\mathsf{H}}$.

Construction 1. *Let λ be the security parameter, $n \in \mathbb{N}$ number of turns, a sender P_A and a receiver P_B. Instantiate the one-way channel scheme with the PPT algorithms $(\mathit{setup}, \mathit{send}, \mathit{ext}, \mathit{turntoken}, \mathsf{valid\text{-}ver}, \mathit{tamper\text{-}ver})$ defined as:*

- $\mathit{setup}(\lambda, \Delta, n) \rightarrow (\mathcal{C}, \mathcal{C}_{priv})$*: to setup the communication channel, P_A parses the security parameter λ, the delay Δ and the number of turns n and executes the algorithm $\mathsf{GenPuz}(\lambda, (n, \Delta))$ as defined in Def. 42 and obtains the n turn puzzle with solution $(\mathbf{y}, \pi)$. Output $(\mathcal{C}, \mathcal{C}_{priv})$ as $(\mathbf{y}, \pi)$;*
- $\mathit{send}(\mathcal{C}_{priv}, m, v, \mathsf{t}) \rightarrow (\xi, \mathsf{aux})$*: to send a message m with validity v in the turn $\mathsf{t} < n$, P_A parses the private channel information $\mathcal{C}_{priv} = \pi$, and compute the values $h_{\mathsf{t}-1} := H(M_{\mathsf{t}-1}, m, \pi_{\mathsf{t}-1})$, $\xi := H(m, \pi_{\mathsf{t}})$ and $\sigma := H(m, \xi, \pi_{\mathsf{t}+1})$ where $M_{\mathsf{t}-1}$*

is the ordered list of valid messages in the turn $(\mathsf{t}-1)$*, together with validity proof and auxiliary information. The sending algorithm outputs, if* $v=1$*, the message correctness proof* ξ *and the channel auxiliary information* $\mathsf{aux}=(h_{\mathsf{t}-1},\sigma)$*, otherwise random values* (ξ,aux) *different from the correct ones.*

- $\mathsf{turntoken}(\mathcal{C},\mathsf{t},\{x_0,\ldots,x_{\mathsf{t}-1}\})\rightarrow x_{\mathsf{t}}$*: this algorithm is executed by the receiver* P_B *at the beginning of turn* t*. It parses the channel* $\mathcal{C}=\mathbf{y}$ *and continually executes* $\mathsf{SolPuz}(\mathbf{y})$ *by considering that every* $\pi_i:=x_i$ *for the* t *partial solution. After* Δ *amount of time, the output of the algorithm is* $x_{\mathsf{t}}:=\pi_{\mathsf{t}}$*.*
- $\mathsf{valid\text{-}ver}(\mathcal{C},\mathsf{t},m,\xi,x_{\mathsf{t}})\rightarrow\{0,1\}$*: at the end of the* t*-th turn, the validity verification takes as input a message* m *and its proof* ξ *and the turn token* $x_{\mathsf{t}}=\pi_{\mathsf{t}}$*. Output* 1 *if the equality* $H(m,\pi_{\mathsf{t}})\stackrel{?}{=}\xi$ *holds. Otherwise,* 0*;*
- $\mathsf{tamper\text{-}ver}(\mathcal{C},\mathsf{t},M_{\mathsf{t}-1},m,\mathsf{aux},\xi)\rightarrow\{0,1\}$*: during the* t*-th turn, the receiver* P_B *verify the correctness of the ordered* $(\mathsf{t}-1)$*-th block* $M_{\mathsf{t}-1}$ *which contains the previously valid ordered messages* $\{m_i\}_{i=1}^{j_{\mathsf{t}-1}}$ *for some* $j_{\mathsf{t}-1}\in\mathbb{N}$*, by parsing the auxiliary information as* $\mathsf{aux}=(h_{\mathsf{t}-1},\sigma)$ *and outputs the result of the equality verification* $H(M_{\mathsf{t}-1},m,\pi_{\mathsf{t}-1})\stackrel{?}{=}h_{\mathsf{t}-1}$*.*
- $\mathsf{ext}(\mathcal{C},\mathcal{C}_{priv},\mathsf{t})\rightarrow x_{\mathsf{t}}$*: the extraction algorithm takes as input the public channel* $\mathcal{C}$*, the private channel* $\mathcal{C}_{priv}=\pi$ *and a turn* $\mathsf{t}\leq n$ *and outputs* $x_{\mathsf{t}}=\pi_{\mathsf{t}}$*;*
- $\mathsf{backward\text{-}ver}(\mathcal{C},\mathsf{t},M_{\mathsf{t}-1},l)\rightarrow\{0,1\}$*: the algorithm takes as input the public channel* $\mathcal{C}$*, the current turn* t*, the previous ordered block* $M_{\mathsf{t}-1}$*, of accepted message* m_i *for* $i\in[1,j_{\mathsf{t}-1}]$*, and an index* l *such that* $m^\star$ *is the* l*-th message in the block* $m^\star=m_l\in M_{\mathsf{t}-1}$ *with auxiliary information* $\mathsf{aux}^\star=\mathsf{aux}_l=(h_{\mathsf{t}-2}{}^\star,\sigma^\star)$*.* $\mathsf{backward\text{-}ver}$ *computes* $\xi^\star=H(m^\star,\pi_{\mathsf{t}-1})$ *and outputs if* $H(m^\star,\xi^\star,\pi_{\mathsf{t}})\stackrel{?}{=}\sigma^\star$*. The* $\mathsf{backward\text{-}ver}$ *algorithm verifies at the end of turn* t *if the message* $m^\star$ *is a correct message for the block* $M_{\mathsf{t}-1}$*.*

Proposition 11. *The proposed one-way channel instantiation of Construction 1 achieves channel correctness as stated in Def. 44.*

Proof. Consider a turn $\mathsf{t}\leq n$ for an n-turn one-way channel defined by executing $(\mathcal{C},\mathcal{C}_{priv})\leftarrow\mathsf{setup}(\lambda,\Delta,n)$. For any message m with validity v, compute the value $\mathsf{send}(\mathcal{C}_{priv},\mathsf{m},v,\mathsf{t})\rightarrow(\xi,(\mathsf{h}_{\mathsf{t}-1},\sigma))$ of which ξ is either $\mathsf{H}(\mathsf{m},\pi_{\mathsf{t}})$ if v=1 otherwise it is an incorrect value. Furthermore execute $\mathsf{ext}(\mathcal{C},\mathcal{C}_{priv},\mathsf{t})\rightarrow\pi_{\mathsf{t}}$. By definition, we have that $\mathsf{valid\text{-}ver}(\mathcal{C},\mathsf{t},\mathsf{m},\xi,\pi_{\mathsf{t}})$ outputs as validity the equality of $\mathsf{H}(\mathsf{m},\pi_{\mathsf{t}})\stackrel{?}{=}\xi$ which is 1, when correctly computed, and 0 otherwise. Assume the existence of an adversary $\mathcal{A}$ able to break the correctness property with some non-negligible probability $\nu>0$, *i.e.* $\mathcal{A}$ is able to produce an invalid pair $(\mathsf{m}^\star,\pi_{\mathsf{t}}{}^\star)$ such that $\mathsf{valid\text{-}ver}(\mathcal{C},\mathsf{t},\mathsf{m}^\star,\xi,\pi_{\mathsf{t}}{}^\star)=1$ for some given digest ξ with probability ν. Let $\epsilon_{\mathsf{H}.pre}$ be the assumed negligible probability of finding a digest pre-image for H of ξ. Construct an adversary $\mathcal{B}$ that reduce the pre-image computation to the one-way correctness by querying $\mathcal{A}$ for a pair $(\mathsf{m}^\star,\pi_{\mathsf{t}}{}^\star)$ for the digest ξ. $\mathcal{B}$ outputs as pre-image the value $(\mathsf{m}^\star,\pi_{\mathsf{t}}{}^\star)$. We conclude that:

$$\nu=\mathsf{Pr}\left[\mathsf{valid\text{-}ver}(\mathcal{C},\mathsf{t},\mathsf{m},\xi,x_{\mathsf{t}})\neq v\;\middle|\;\begin{array}{l}\mathsf{setup}(\lambda,\Delta,n)\rightarrow(\mathcal{C},\mathcal{C}_{priv});\\ \mathsf{send}(\mathcal{C}_{priv},\mathsf{m},v,\mathsf{t})\rightarrow(\xi,\mathsf{aux});\\ \mathsf{ext}(\mathcal{C},\mathcal{C}_{priv},\mathsf{t})\rightarrow x_{\mathsf{t}};\end{array}\right]\leq\epsilon_{\mathsf{H}.pre}$$

which is absurd. Thus proving the correctness property. □

Proposition 12. *The proposed one-way channel instantiation of Construction 1 achieves sequentiality as stated in Def. 45.*

Proof. Consider the sequentiality game $\mathsf{Game}_{seq}^{\mathcal{A},\Delta}(\lambda,\mathsf{t},n)$ in which the challenger generates the communication channel $(\mathcal{C},\mathcal{C}_{priv})$ and let $\mathsf{t}\leq n$ be an arbitrary turn in which

the adversary is challenged. The challenger chooses an arbitrary message m and validity $v \leftarrow \{0,1\}$ and executes $\mathsf{send}(\mathcal{C}_{priv}, \mathsf{m}, v, \mathsf{t}) \rightarrow (\xi, \mathsf{aux})$. The adversary $\mathcal{A}$ wins the game if the output $v^* \leftarrow \mathcal{A}\big(\mathcal{C}, \mathsf{t}, m, \xi, \mathsf{aux}, \{x_i\}_{i=1}^{t-1}\big)$ is the challenger's chosen validity v **and** the execution time for the adversary is bounded as $\mu(\mathcal{A}) < \Delta$. $\mathcal{A}$ can therefore be used by an adversary $\mathcal{B}$ to reduce the Δ-delay property for the Δ-delay hash function to the one-way sequentiality game. Briefly, if we assume $\mathcal{A}$ to have a non-negligible probability to compute v, $\mathcal{B}$ is able to break the Δ-delay property which is assumed to be hard. □

Proposition 13. *The proposed one-way channel instantiation of Construction 1 achieves last turn tamper resistance as stated in Def. 46.*

Proof. Consider a communication between two honest parties to generate the blocks M_i for $i \in \{1, \ldots, t-1\}$ where $t \leq n$ is the turn in which the adversary $\mathcal{A}$ will output the tuple $(\mathsf{M}^\star, \mathsf{m}^\star, (\mathsf{h}^\star, \sigma^\star), \xi^\star)$, which contains a tampered block for the turn $t-1$, a tampered message and the related auxiliary information and the tampered validity proof. Observe that the verification algorithm will compute $\mathsf{tamper\text{-}ver}(\mathcal{C}, t, \mathsf{M}_{t-1}, \mathsf{m}^\star, (\mathsf{h}^\star, \sigma^\star), \xi^\star)$ with the correct block, which will verify the equality of $\mathsf{H}(\mathsf{M}_{t-1}, \mathsf{m}^\star, \pi_{t-1}) \stackrel{?}{=} \mathsf{h}^\star$. Obviously, $\mathcal{A}$ can always generate, for any messages, correctly evaluated digests. However, in order to correctly consider it a tamper, the adversarial tamper must verify the algorithm with the tampered block. Then, to allow the existence of two correct but different block visions, *i.e.* formally $\mathsf{tamper\text{-}ver}(\mathcal{C}, t, \mathsf{M}^\star, \mathsf{m}^\star, (\mathsf{h}^\star, \sigma^\star), \xi^\star)$, which is equivalent to $\mathsf{H}(\mathsf{M}^\star, \mathsf{m}^\star, \pi_{t-1}) \stackrel{?}{=} \mathsf{h}^\star$. Assume by absurd that such $\mathcal{A}$ exists and outputs correct tampers with non-negligible probability $\nu > 0$. Intuitively, construct an adversary $\mathcal{B}$ that reduce the second pre-image computation to the one-way tampering by querying $\mathcal{A}$. $\mathcal{A}$ must provide a second pre-image $(\mathsf{M}^\star, \mathsf{m}^\star)$ of the digest $\mathsf{h}^\star$ obtained from $(\mathsf{M}_{t-1}, \mathsf{m}^\star)$. Thus, $\mathcal{B}$ outputs a second pre-image of $\mathsf{h}^\star$ with probability $\nu \leq \epsilon_{\mathsf{H}.2pre}$ which is assumed to be negligible. □

Proposition 14. *Consistency ⇔ last turn tamper resistant and correctness.*

Proof. The proof of this proposition is trivial. Our definition of consistency is similar to the definition of tamper resistance where we additionally require the tampered block to be formed only by *correct* messages. Therefore, a consistent channel is trivially correct and tamper resistant. For the opposite implication, assume that the channel is non-consistent, *i.e.* an adversary can compute a wrong message view in a specific turn. This is true if and only if the adversary can create a correct tamper block which contains at least a wrong message-proof ξ and auxiliary information tuple aux. This implies that a non-consistent channel allows to break the correctness and tamper resistance property. □

3.3 Two-Way TBCC

In this section, we instantiate a two-way TBCC and explain how to correctly realise the *recovery procedure*, *i.e.* a procedure executed between the parties that allows them to force the communication's correctness and coherence.

Consider the parties P_A and P_B and let both independently setup the consistent one-way channel of Construction 1 which casts them both as receiver and sender into two independent channels each. Both parties can send a message to the other one in the channel they created. Concurrently, each party keeps track of its local turn to receive and check messages by *(1)* continuously executing $\mathsf{turntoken}$ and *(2)* keeping of the previously generated turn tokens x_i for $i \leq \mathsf{t}$.

Protocol 3 (The Two-Way TBCC Protocol)**.** *Given two parties P_A and P_B, an integer value n and real non-zero value Δ, define the (Two-Way) TBCC across n turns with delay Δ with the procedures:*

- **Setup:** *on input the security parameter λ, P_A (respectively P_B) executes the algorithm $\mathsf{setup}(\lambda, \Delta, n)$, obtains $(\mathcal{C}_A, \mathcal{C}_{A,priv})$, and sends $\mathcal{C}_A$ to P_B, which replies with $\mathcal{C}_B$. P_A outputs the two-way TBCC channel information $(\mathcal{C}_A, \mathcal{C}_B)$, along with its respective private information $\mathcal{C}_{priv}$ and P_A performs $\mathsf{turntoken}(\mathcal{C}_B, 1, x_{B,0})$;*
- **Local Turn (analogously for P_B):** *on receiving a call to this procedure, P_A returns the current local turn t corresponding to the last computed $x_{P_B,\mathsf{t}}$;*
- **Send Message (analogously for P_B):** *on a given local turn t, when P_A receives the input (m, v), it executes $\mathsf{send}(\mathcal{C}_{A,priv}, \mathsf{m}, v, \mathsf{t}) \rightarrow (\xi, \mathsf{aux})$ where the previous block digest is computed as $\mathsf{h}_{\mathsf{t}-1} := H(\mathsf{M}_{\mathsf{t}-1}, \mathsf{m}, \pi^{P_A}_{\mathsf{t}-1}, \pi^{P_B}_{\mathsf{t}-1})$, and sends $(\mathsf{m}, \xi, \mathsf{aux})$ to P_B;*
- **Reveal Validity (analogously for P_A):** *at the end of the local turn t,* i.e. *when the algorithm $\mathsf{turntoken}(\mathcal{C}_A, \mathsf{t}, \{x_{A,0}, \ldots, x_{A,\mathsf{t}-1}\})$ outputs the token $x_{A,\mathsf{t}}$, P_B executes $\mathsf{valid\text{-}ver}(\mathcal{C}_A, \mathsf{t}, m_i, \xi_i, x_{A,\mathsf{t}}) \rightarrow v_i$, and outputs the block of* both the parties *valid messages $\mathsf{M}_\mathsf{t}=\{(m_i, \xi_i, \mathsf{aux}_i)\}_i$ along with the turn token t whenever $v_i=1$. Furthermore, for all the messages m_i, $\mathsf{tamper\text{-}ver}(\mathcal{C}_A, \mathsf{t}, \mathsf{M}_{\mathsf{t}-1}, \mathsf{m}_i, \mathsf{aux}_i, \xi_i)$ is executed and if any result is 0, abort the communication. If $\mathsf{t}+1 > n$, then output* CLOSE *and stop. Otherwise, execute $\mathsf{turntoken}(\mathcal{C}_A, \mathsf{t}+1, \{x_{A,0}, \ldots, x_{A,\mathsf{t}}\})$.*

Remark 15. *The TBCC protocol naturally extends the one-way properties of correctness and tamper resistance to the two-way channel. For example, if the two-way channel is tamperable, it means the adversary can tamper at least one direction of the communication channel. In other words, tamper the one-way channel.* Mutatis mutandis *the same is true for the correctness property.*

Turn Synchronization and Consistency. When considering the two-way protocol by instantiating two one-way turn based schemes, an additional problem that naturally arises is *turn synchronization between the parties.* Consider the parties P_A and P_B communicating using Proto. 3 which depends on the specific one-way channels $\mathcal{C}_A$ and $\mathcal{C}_B$. The specific channel turn is identified by the input of the algorithm turntoken which are, *almost surely*, never synchronized, *i.e.* the outputs are disclosed in different moments. This timing lack creates a problem in which a message m might be seen in turn t by P_A and in turn $(\mathsf{t}+1)$ by P_B. We capture this idea by formalizing the *turn synchronization* property.

Definition 48 (Turn Synchronization)**.** *Let P_A and P_B be parties communicating over the two-way TBCC. The TBCC channel $(\mathcal{C}_A, \mathcal{C}_B)$ is **turn-consistent** if both players have a **unique and equal** way to decide in which turn the message m belongs even then the **local turns** of the two parties are different.*

The TBCC without turn synchronization cannot achieve communication consistency since the parties might disagree in which block M the message m belongs, making it unlikely to create an unique communication history. Intuitively, achieving sequentiality means that the turntoken algorithm is defining a *"clock"*, *i.e.* sequential *"ticks"* distanced by some amount of time, while being desynchronized means that the parties have *"different clocks"* where one of the two is always *"late"*. We prove that if we have a sequential one-way scheme, then there exists a natural way to achieve turn-consistency by letting the parties **avoid communicating** in between the *"ticks"* thus allowing the *"late clock"* to sync.

Proposition 15. *Let P_A and P_B be parties communicating via the two-way TBCC protocol, constructed from a sequential one-way scheme as in Def. 45. The strategy*

of (i) dropping communicated messages during de-synchronization, i.e. *the local turn between the parties is different; and (ii)* globally *advance the turn whenever both parties have the same local turn; allows turn-consistency as in Def. 48.*

Recovery Procedure. We consider the existence of a *recovery procedure* that should be executed whenever a party spots a possible communication tamper and, instead of directly aborting the protocol, the two parties try to find a common correct message block. In other words, the algorithm tamper-ver from Construction 1 takes as input the last block views M_{P_A} and M_{P_B} that the two parties have and either outputs a commonly agreeable block M or aborts.

Definition 49 (Recovery)**.** *Define the recovery procedure for Proto. 3 as the procedure executed during turn* $\mathsf{t} \leq n$ *by* P_A *(resp.* P_B*) whenever the tamper verification* $\mathsf{tamper\text{-}ver}(\mathcal{C}, \mathsf{t}, M_{\mathsf{t}-1}, m, aux, \xi)$ *is equal* 0 *and defined as:*

- **Recovery***:* P_A *sends its view* $M^A_{\mathsf{t}-1}$ *to* P_B *from whom it receives the view* $M^B_{\mathsf{t}-1}$ *which is a ordered list of messages* $\{m_i\}_{i=1}^{j_{\mathsf{t}-1}}$ *and, additionally, for every message the received auxiliary information* σ*. After identifying the set of indexes* I *where the views differ, for each index* $l \in I$*, if the message* m_l *is a message from* P_B*, then* P_A *executes* $\mathsf{backward\text{-}ver}(\mathcal{C}_B, \mathsf{t}, M^B_{\mathsf{t}-1}, l)$*, otherwise* P_B *will compute* $\mathsf{backward\text{-}ver}(\mathcal{C}_A, \mathsf{t}, M^A_{\mathsf{t}-1}, l)$*. Either the case, if the result is* 1*, both parties are forced to use the message* m_l *resolving the discrepancy and saving the result into the same resolved block* $M_{\mathsf{t}-1}$*. Otherwise, if there exists an index for which the result is* 0*, the communication is aborted.*

The spirit of the TBCC is *"if anything seems wrong, abort!"*. This forces the parties to behave honestly otherwise nothing can be achieved, meaning there can never exist **two** different correct views. During the recovery procedure, the communication is paused and completely verified and fixed before continuing and, if necessary, aborted because it is unrecoverable. The receiver must be aware and promptly alert the sender if h_{i-1} is wrong and, if it is the case, only the receiver can force the sender to adopt a specific message m_i by exhibiting the received proof σ_i, only computable by the sender.

Formally, suppose P_A and P_B are correctly communicating until the i-th turn, *i.e.* all the blocks until M_{i-1} are consistent. P_A sends $\left(\mathsf{h}^A_{i-1}, \mathsf{m}^A_i, \xi^A_i, \sigma^A_i\right)$ and P_B does the same with the message m^B_i. Let us suppose that the values $\{\xi^A_i, \xi^B_i\}$ are correct otherwise the messages will be discarded by valid-ver. Thus the correct next block is $\mathsf{M}_i = \{\mathsf{m}^A_i, \mathsf{m}^B_i\}$. Whenever the turn $(i+1)$ starts, P_B and P_A must share the block digests h^A_i and h^B_i and suppose they are not equal.

The recovery procedure is executed and P_B will publish the block-view $\{\mathsf{m}^A_i, \mathsf{m}^B_i\}$, respectively P_A must do the same, and there must be at least a different message pair, *w.l.o.g.* suppose it is message m^A_i and $\mathsf{m}^{A^\star}$. Since this is the message that Alice sent, in the recovery, we will just consider Bob's view $\mathsf{m}^{A^\star}$ with received auxiliary information $\sigma^{A^\star}$ which Bob cannot correctly forge by assumption, *i.e.* he cannot produce a correct valid pair. Therefore P_B can only re-publish what P_A sent **or** abort the communication. Regardless of P_B's maliciousness, he is unable to modify Alice's messages and therefore the procedure continues only if $\sigma^{A^\star}$ is correctly computed by P_A. In the case that Bob's message m^B_i is different, Alice's vision is considered. If Bob is honest, the previous discussion applies for Alice. Otherwise, Bob might try to force the acceptance of a different pair $(\mathsf{m}^{B^\star}, \sigma^{B^\star})$. Since his vision during recovery is not considered, he must have sent the tampered values $(\mathsf{m}^{B^\star}, \sigma^{B^\star})$ *before* but if this is the case, either Alice is presenting the tampered pair $(\mathsf{m}^{B^\star}, \sigma^{B^\star})$, which makes the pair not longer a tamper, since it is correctly received by Alice and not later modified, or by sending an incorrect pair that will lead to aborting the communication. *Mutatis mutandis*, the same is

true when switching P_A and P_B roles. If everything is correct, the block vision is consolidated, communication can resume and the only real cost is that both P_A and P_B lost a single turn.

4 Collectively Flipping Coins over the TBCC

In this section, we sketch a protocol that allows two parties to *collectively flip a coin* which allows them to commonly create a random string. Our TBCC protocol is constructed from time-lock puzzles which are used in similar applications, as:

- a user can create encrypted time capsule, *i.e.* an encrypted message that is meant to only be decryptable after a designed amount of time;
- a user can provide a signature that can only be verified in the future.

As discussed by Rivest *et al.* [RSW96], these are founded on the concept of releasing a *timed commitment* that can be decommitted after a specific amount of time.

The provided coin-flip solution is *simplistic* and it has the main goal of showing the TBCC's expressiveness/potentiality. To provide a formal security analysis, TBCC must be proven secure against active adversaries and general protocol's composability which, as previously assumed, are left open for future research.

Flipping Coins over TBCC. The underlying idea is that two parties, communicating over a TBCC's instance, are able to *jointly* flip a coin by both time-committing to some randomness which is later revealed and used to compute the coin result. By repeatedly flipping coins, the results produce a random string which is guaranteed to be consistent since communicated over TBCC.

Let us provide a formalisation of the collectively coin-flip between Alice P_A and Bob P_B. These protocols are defined by a set of choices Σ and a set of rules that allows to determine the *result* between any two choices, denoted with the function $\mu(\cdot,\cdot)$. Formally, the collective coin-flip protocol is therefore defined as:

(a) P_A and P_B set up the two-way TBCC protocol of Proto. 3 and obtain the public channel $\mathcal{C} = (\mathcal{C}_{P_A}, \mathcal{C}_{P_B})$;
(b) In the current turn, P_A selects its choice $a \in \Sigma$ and sends on $\mathcal{C}$ as a valid message, *i.e.* P_A execute the sending procedure with the message $(a, 1)$. For each other choice $a^\star \in \Sigma$, P_A sends the non-valid message $(a^\star, 0)$. Respectively, P_B sends his valid and invalid messages;
(c) At the end of the turn, P_A computes the validity of P_B's received messages and obtains b. Respectively for P_B;
(d) Both the parties compute $\mu(a, b)$ and, if necessary, repeat the game. If the channel loses consistency, *i.e.* one of the party tries to tamper the results, the communication is aborted;
(e) The random string is obtained by concatenating several consecutive results of the consistent channel.

The *"commit-decommit"* phase created by the turn token is key to allow a fair-play since, for example, if P_A knows P_B's choice b in advance, she can select a winning choice $a^\star$. Furthermore, ϕ must be defined even in the case of one party not participating in the round or it tries to cheat by proposing multiple choices. We are now left to define the choice's set Σ and the rule's map $\mu(\cdot,\cdot)$. Σ contains the choices head and tail, respectively 1 and 0 and, additionally, a special element x that represents any non-correct choice, *i.e.* a party does not correctly participate in the game. Define the map μ as $\mu(a, b) = a \oplus b$, *i.e.* the xor between the inputs where the special element is mapped as $\mu(\mathsf{x}, a) = \mu(a, \mathsf{x}) = a$ for each $a \in \Sigma$ and we consider a special state X used to denote that both player wrongly participated in the flipping, *i.e.* $\mu(\mathsf{x}, \mathsf{x}) = \mathsf{X}$. In a nutshell, $\mu(a, b)$ computes the xor of both the parties inputs whenever they are correctly participating in the coin-flip. Complementary, if both the parties wrongly flip the coin, $\mu(\mathsf{x}, \mathsf{x})$ returns

that the coin is in a *"draw position"* with *"no winner"*. Whenever a party, *e.g.* P_A, wrongly participates in the protocol, $\mu(\mathsf{x}, b)$ awards the other party P_B for correctly behaving and let P_B's choice be the final result. This forces the parties to correctly behave to avoid the other party highly influence the coin-flip. For example, suppose that P_A selects 1 as her first choice and sends to P_B the TBCC messages $(1, 1)$ and $(0, 0)$ during the current turn. By the sequentiality property, P_B is unable to discover *"which message is the valid one"* and therefore has no advantage and must therefore provide his own choice, *w.l.o.g.* let P_B choose 0. At the end of the turn, the valid messages are maintained thus the block will contain P_A's message 1 and P_B's one 0. Both the parties can now compute $\mu(1, 0) = 1$ and acknowledge that the coin flip is 1. The TBCC protocol guarantees communication coherence which implies that, whenever repeating the game, both the parties **must** accept the previous communication transcription. In other words, *while* communicating over $\mathcal{C}$, P_A and P_B cannot modify the output of the different rounds played. This means that if the result is 1, in the next round P_B cannot pretend a different outcome and must accept it if he wants to participate in the next round. The game output's transcript can be seen as a random string between P_A and P_B which cannot be tampered with by a malicious adversary. Additionally, every time the adversary is caught tampering or deny the communication, the whole protocol is terminated making it impossible for the adversary to gain any relevant advantage.

We must point out that our protocol **does not** approximate a *public coin flip* one which can be used to generate the common reference string model. In the public coin-flip protocol, the two parties obtain a random coin-flip *without* introducing their own personally sampled randomness. For this reason, our protocol can be used to approximate an empirical version of the common reference string model in which the parties *actively collaborate* to sample a random string.

Bibliography

[Adl83] Leonard M. Adleman. Implementing an Electronic Notary Public. In *Advances in Cryptology*, 1983.

[AFS05] Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. A Family of Fast Syndrome Based Cryptographic Hash Functions. In *Progress in Cryptology – Mycrypt 2005*, 2005.

[AG17] Hunt Allcott and Matthew Gentzkow. Social Media and Fake News in the 2016 Election. *J. Econ. Perspect.*, 31(2), May 2017.

[AGM+13] Joseph A. Akinyele, Christina Garman, Ian Miers, Matthew W. Pagano, Michael Rushanan, Matthew Green, and Aviel D. Rubin. Charm: A framework for rapidly prototyping cryptosystems. *J Cryptogr Eng*, 3(2), June 2013.

[AGP16] Pablo Daniel Azar, Shafi Goldwasser, and Sunoo Park. How to Incentivize Data-Driven Collaboration Among Competing Parties. In *ITCS*, 2016.

[AGS11] C. Aguilar, P. Gaborit, and J. Schrek. A new zero-knowledge code based identification scheme with reduced communication. In *2011 IEEE Information Theory Workshop*, October 2011.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, July 1996.

[Alp14] Ethem Alpaydin. *Introduction to Machine Learning*. Third edition edition, 2014.

[AT17] Joël Alwen and Björn Tackmann. Moderately Hard Functions: Definition, Instantiations, and Applications. In *TCC*, 2017.

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable Delay Functions. In *CRYPTO*, volume 10991. 2018.

[BBCD20] Anubhab Baksi, Jakub Breier, Yi Chen, and Xiaoyang Dong. Machine learning assisted differential distinguishers for lightweight ciphers (extended version). 2020.

[BBF19] Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains. In *CRYPTO*, 2019.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, January 2012.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, June 2013.

[BCF17] Manuel Barbosa, Dario Catalano, and Dario Fiore. Labeled Homomorphic Encryption. In *Computer Security – ESORICS 2017*, 2017.

[BCS19] Carlo Brunetta, Marco Calderini, and Massimiliano Sala. On hidden sums compatible with a given block cipher diffusion layer. *Discrete Math.*, 342(2), February 2019.

[BDD+20] Carsten Baum, Bernardo David, Rafael Dowsley, Jesper Buus Nielsen, and Sabine Oechsner. TARDIS: Time And Relative Delays In Simulation. Technical Report 537, 2020.

[BDL+16] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More Efficient Commitments from Structured Lattice Assumptions. Technical Report 997, 2016.

[BDLM17] Carlo Brunetta, Christos Dimitrakakis, Bei Liang, and Aikaterini Mitrokotsa. A Differentially Private Encryption Scheme. In *Information Security*, 2017.

[BEB13] Robert G Brown, Dirk Eddelbuettel, and David Bauer. Dieharder: A random number test suite. *Open Source Softw. Libr.*, 2013.

[Bei11] Amos Beimel. Secret-Sharing Schemes: A Survey. In *Coding and Cryptology*, 2011.

[Ben87] Josh Cohen Benaloh. Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret (Extended Abstract). In *Advances in Cryptology — CRYPTO' 86*, 1987.

[BF14] Mihir Bellare and Georg Fuchsbauer. Policy-Based Signatures. In *Public-Key Cryptography – PKC 2014*, 2014.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional Signatures and Pseudorandom Functions. In *Public-Key Cryptography – PKC 2014*, 2014.

[BGI17] Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-Based Secure Computation: Optimizing Rounds, Communication, and Computation. In *Advances in Cryptology – EUROCRYPT 2017*, 2017.

[BGJ+16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-Lock Puzzles from Randomized Encodings. In *ITCS*, 2016.

[BGM16] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Cryptocurrencies Without Proof of Work. In *FC*, 2016.

[BIK+17] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, October 2017.

[BK15] Elaine B. Barker and John M. Kelsey. Recommendation for Random Number Generation Using Deterministic Random Bit Generators. Technical Report NIST SP 800-90Ar1, National Institute of Standards and Technology, June 2015.

[BK16] Elaine Barker and John Kelsey. Recommendation for Random Bit Generator (RBG) Constructions. Technical Report NIST Special Publication (SP) 800-90C (Draft), National Institute of Standards and Technology, April 2016.

[BKLP15] Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. Efficient Zero-Knowledge Proofs for Commitments from Learning with Errors over Rings. In *Proceedings, Part I, of the 20th European Symposium on Computer Security – ESORICS 2015 - Volume 9326*, 2015.

[BLM18] Carlo Brunetta, Bei Liang, and Aikaterini Mitrokotsa. Lattice-Based Simulatable VRFs: Challenges and Future Directions. *J. Internet Serv. Inf. Secur. JISIS*, 8(4), November 2018.

[BLM19] Carlo Brunetta, Bei Liang, and Aikaterini Mitrokotsa. Code-Based Zero Knowledge PRF Arguments. In *Information Security*, 2019.

[BLMR13] Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key Homomorphic PRFs and Their Applications. In *Advances in Cryptology – CRYPTO 2013*, 2013.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. *J. Cryptol.*, 17(4), September 2004.

[BMS16] Michael Backes, Sebastian Meiser, and Dominique Schröder. Delegatable Functional Signatures. In *Public-Key Cryptography – PKC 2016*, 2016.

[BMv78] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (Corresp.). *IEEE Trans. Inform. Theory*, 24(3), May 1978.

[BN00] Dan Boneh and Moni Naor. Timed Commitments. In *CRYPTO*, 2000.

[BNO11] Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed Private Data Analysis: On Simultaneously Solving How and What. *ArXiv11032626 Cs*, March 2011.

[BP21] Carlo Brunetta and Pablo Picazo-Sanchez. Modelling cryptographic distinguishers using machine learning. *J. Cryptogr. Eng.*, July 2021.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom Functions and Lattices. In *Advances in Cryptology – EUROCRYPT 2012*, volume 7237. 2012.

[Bri90] Ernest F. Brickell. Some Ideal Secret Sharing Schemes. In *Advances in Cryptology — EUROCRYPT '89*, 1990.

[BRS+10] Lawrence Bassham, Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, N. Heckert, and James Dray. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Technical Report NIST Special Publication (SP) 800-22 Rev. 1a, National Institute of Standards and Technology, April 2010.

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptology*, 4(1), January 1991.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.*, 43(2), 2014.

[CG86] Alfredo Capelli and Giovanni Garbieri. *Corso Di Analisi Algebrica: 1: Teorie Introduttorie*, volume 1. 1886.

[CGG07] Pierre-Louis Cayrel, Philippe Gaborit, and Marc Girault. Identity-Based Identification and Signature Schemes Using Correcting Codes. In *WCC*, volume 2007, 2007.

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, October 1985.

[Cha95] Florent Chabaud. On the security of some cryptosystems based on error-correcting codes. In *Advances in Cryptology — EUROCRYPT'94*, 1995.

[CKP+20] S. Cohney, A. Kwong, S. Paz, D. Genkin, N. Heninger, E. Ronen, and Y. Yarom. Pseudorandom black swans: Cache attacks on CTR_DRBG. In *S&P*, May 2020.

[CL07] Melissa Chase and Anna Lysyanskaya. Simulatable VRFs with Applications to Multi-theorem NIZK. In *Advances in Cryptology - CRYPTO 2007*, August 2007.

[CLZ12] Rafik Chaabouni, Helger Lipmaa, and Bingsheng Zhang. A Non-interactive Range Proof with Constant Communication. In *Financial Cryptography and Data Security*, 2012.

[Con18] A. Connolly. Freedom of Encryption. *IEEE Secur. Priv.*, 16(1), January 2018.

[Cou16] Council of the European Union, European Parliament. *Regulation (EU) 2016/679 (General Data Protection Regulation)*. 2016.

[CPSV16] Michele Ciampi, Giuseppe Persiano, Luisa Siniscalchi, and Ivan Visconti. A Transform for NIZK Almost as Efficient and General as the Fiat-Shamir Transform Without Programmable Random Oracles. In *Theory of Cryptography*, 2016.

[CRRV17] Ran Canetti, Srinivasan Raghuraman, Silas Richelson, and Vinod Vaikuntanathan. Chosen-Ciphertext Secure Fully Homomorphic Encryption. In *Public-Key Cryptography – PKC 2017*, 2017.

[CV07] Dario Catalano and Ivan Visconti. Hybrid Commitments and Their Applications to Zero-knowledge Proof Systems. *Theor Comput Sci*, 374(1-3), April 2007.

[CVEYA11] Pierre-Louis Cayrel, Pascal Véron, and Sidi Mohamed El Yousfi Alaoui. A Zero-Knowledge Identification Scheme Based on the q-ary Syndrome Decoding Problem. In *Selected Areas in Cryptography*, 2011.

[CZD+19] Chengjun Cai, Yifeng Zheng, Yuefeng Du, Zhan Qin, and Cong Wang. Towards Private, Robust, and Verifiable Crowdsensing Systems via Public Blockchains. *IEEE Trans. Dependable and Secure Comput.*, 2019.

[DGKR18] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros Praos: An Adaptively-Secure, Semi-synchronous Proof-of-Stake Blockchain. In *Advances in Cryptology – EUROCRYPT 2018*, 2018.

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Trans. Inf. Theory*, 22(6), 1976.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*, volume 3876. 2006.

[DNR04] Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing Encryption Schemes from Decryption Errors. In *Advances in Cryptology - EUROCRYPT 2004*, 2004.

[DNS04] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. *J. ACM*, 51(6), November 2004.

[DS06] A.D. Dileep and C.C. Sekhar. Identification of Block Ciphers using Support Vector Machines. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, July 2006.

[Dwo06] Cynthia Dwork. Differential Privacy. In *Automata, Languages and Programming*, 2006.

[EED08] Khaled El Emam and Fida Kamal Dankar. Protecting Privacy Using k-Anonymity. *J Am Med Inf. Assoc*, 15(5), 2008.

[ElG85] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology*, 1985.

[ELL+15] Martianus Frederic Ezerman, Hyung Tae Lee, San Ling, Khoa Nguyen, and Huaxiong Wang. A Provably Secure Group Signature Scheme from Code-Based Assumptions. In *Advances in Cryptology – ASIACRYPT 2015*, 2015.

[ETLP13] Z. Erkin, J. R. Troncoso-pastoriza, R. L. Lagendijk, and F. Perez-Gonzalez. Privacy-preserving data aggregation in smart metering systems: An overview. *IEEE Signal Process. Mag.*, 30(2), March 2013.

[EYACM11] Sidi Mohamed El Yousfi Alaoui, Pierre-Louis Cayrel, and Meziani Mohammed. Improved Identity-Based Identification and Signature Schemes Using Quasi-Dyadic Goppa Codes. In *Information Security and Assurance*, 2011.

[FFKB17] Andreas Fischer, Benny Fuhry, Florian Kerschbaum, and Eric Bodden. Computation on Encrypted Data using Data Flow Authentication. *CoRR*, abs/1710.00390, 2017.

[FG12] Dario Fiore and Rosario Gennaro. Publicly Verifiable Delegation of Large Polynomials and Matrix Computations, with Applications. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, 2012.

[FGJS17] Nelly Fazio, Rosario Gennaro, Tahereh Jafarikhah, and William E. Skeith. Homomorphic Secret Sharing from Paillier Encryption. In *Provable Security*, 2017.

[FGP14] Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently Verifiable Computation on Encrypted Data. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[Fis18] Tilo Fischer. Testing Cryptographically Secure Pseudo Random Number Generators with Artificial Neural Networks. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, August 2018.

[FMNP16] Dario Fiore, Aikaterini Mitrokotsa, Luca Nizzardo, and Elena Pagnin. Multi-key Homomorphic Authenticators. In *Advances in Cryptology – ASIACRYPT 2016*, 2016.

[FS87] Amos Fiat and Adi Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology — CRYPTO' 86*, 1987.

[FS96] Jean-Bernard Fischer and Jacques Stern. An Efficient Pseudo-Random Generator Provably as Secure as Syndrome Decoding. In *Advances in Cryptology — EUROCRYPT '96*, 1996.

[GAC18] Francisco-Javier González-Serrano, Adrián Amor-Martín, and Jorge Casamayón-Antón. Supervised machine learning using encrypted training data. *Int. J. Inf. Secur.*, 17(4), 2018.

[Gen09] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD Thesis, Stanford University, 2009.

[GGG17] Zahra Ghodsi, Tianyu Gu, and Siddharth Garg. SafetyNets: Verifiable execution of deep neural networks on an untrusted cloud. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, 2017.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. *J ACM*, 33(4), August 1986.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In *Advances in Cryptology – CRYPTO 2010*, 2010.

[Gil52] E. N. Gilbert. A comparison of signalling alphabets. *Bell Syst. Tech. J.*, 31(3), May 1952.

[GJ11] Flavio D. Garcia and Bart Jacobs. Privacy-Friendly Energy-Metering via Homomorphic Encryption. In *Security and Trust Management*, 2011.

[GK06] S. Dov Gordon and Jonathan Katz. Rational Secret Sharing, Revisited. In *Security and Cryptography for Networks*, 2006.

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin Backbone Protocol: Analysis and Applications. In *EUROCRYPT*, volume 9057. 2015.

[GKM11] Johannes Gehrke, Daniel Kifer, and Ashwin Machanavajjhala. L-Diversity. In *Encyclopedia of Cryptography and Security*. 2011.

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, February 1989.

[GLS07] P. Gaborit, C. Lauradoux, and N. Sendrier. SYND: A Fast Code-Based Stream Cipher with a Security Reduction. In *2007 IEEE International Symposium on Information Theory*, June 2007.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, May 1982.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2), April 1988.

[GNP+15] Sharon Goldberg, Moni Naor, Dimitrios Papadopoulos, Leonid Reyzin, Sachin Vasant, and Asaf Ziv. NSEC5: Provably Preventing DNSSEC Zone Enumeration. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*, 2015.

[Goh19] Aron Gohr. Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning. In *Advances in Cryptology – CRYPTO 2019*, 2019.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, June 2011.

[GW13] Rosario Gennaro and Daniel Wichs. Fully Homomorphic Message Authenticators. In *Advances in Cryptology - ASIACRYPT 2013*, 2013.

[HAP17] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, October 2017.

[Her28] Alex Hern. Fitness tracking app Strava gives away location of secret US army bases. *The Guardian*, 2018.Jan.28.

[HGDM+11] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: A first study. *J Cryptogr Eng*, 1(4), October 2011.

[Hir09] Shoichi Hirose. Security Analysis of DRBG Using HMAC in NIST SP 800-90. In *Information Security Applications*, 2009.

[HMT13] Rong Hu, Kirill Morozov, and Tsuyoshi Takagi. Proof of plaintext knowledge for code-based public-key encryption revisited. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, May 2013.

[HPS14] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. An Introduction to Cryptography. In *An Introduction to Mathematical Cryptography*. 2014.

[HZ19] Xinyi Hu and Yaqun Zhao. Block Ciphers Classification Based on Random Forest. *J. Phys.: Conf. Ser.*, 1168, February 2019.

[JLE14] Zhanglong Ji, Zachary C. Lipton, and Charles Elkan. Differential Privacy and Machine Learning: A Survey and Review. *ArXiv14127584 Cs*, December 2014.

[Jou09] Antoine Joux. *Algorithmic Cryptanalysis*. 2009.

[KK06] Shri Kant and Shehroz S. Khan. Analyzing a class of pseudo-random bit generator through inductive machine learning paradigm. *Intell. Data Anal.*, 10(6), December 2006.

[KKG+09] Shri Kant, Naveen Kumar, Sanchit Gupta, Amit Singhal, and Rachit Dhasmana. Impact of machine learning algorithms on analysis of stream ciphers. In *2009 Proceeding of International Conference on Methods and Models in Computer Science (ICM2CS)*, December 2009.

[KL08] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. 2008.

[KLP07] Yael Tauman Kalai, Yehuda Lindell, and Manoj Prabhakaran. Concurrent Composition of Secure Protocols in the Timing Model. *J Crypto*, 20(4), October 2007.

[KMS14] Jonathan Katz, Andrew Miller, and Elaine Shi. Pseudonymous Broadcast and Secure Computation from Cryptographic Puzzles. Technical Report 857, 2014.

[KMTZ13] Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally Composable Synchronous Computation. In *TCC*, 2013.

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Math. Comput.*, 48(177), January 1987.

[Koz91] John Koza. Evolving a computer program to generate random numbers using the genetic programming paradigm. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991.

[Kra94] Hugo Krawczyk. Secret Sharing Made Short. In *Advances in Cryptology — CRYPTO' 93*, 1994.

[KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *CRYPTO*, volume 10401, 2017.

[KTX08] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently Secure Identification Schemes Based on the Worst-Case Hardness of Lattice Problems. In *Advances in Cryptology - ASIACRYPT 2008*, 2008.

[LABK17] Wenting Li, Sébastien Andreina, Jens-Matthias Bohli, and Ghassan Karame. Securing Proof-of-Stake Blockchain Protocols. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, volume 10436. 2017.

[Lin15] Yehuda Lindell. An Efficient Transform from Sigma Protocols to NIZK with a CRS and Non-programmable Random Oracle. In *Theory of Cryptography*, 2015.

[LLM+16] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Zero-Knowledge Arguments for Matrix-Vector Relations and Lattice-Based Group Encryption. In *Advances in Cryptology – ASIACRYPT 2016*, 2016.

[LLNW17] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-Knowledge Arguments for Lattice-Based PRFs and Applications to E-Cash. In *Advances in Cryptology – ASIACRYPT 2017*, 2017.

[LLNW18] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-Based Zero-Knowledge Arguments for Integer Relations. In *Advances in Cryptology – CRYPTO 2018*, 2018.

[LLV07] N. Li, T. Li, and S. Venkatasubramanian. T-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, April 2007.

[LMA+18] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *25th Annual Network and Distributed System Security Symposium, NDSS*, 2018.

[LMS18] Russell W. F. Lai, Giulio Malavolta, and Dominique Schröder. Homomorphic Secret Sharing for Low Degree Polynomials. In *Advances in Cryptology – ASIACRYPT 2018*, 2018.

[LS07] Pierre L'Ecuyer and Richard Simard. TestU01: A C library for empirical testing of random number generators. *ACM Trans. Math. Softw.*, 33(4), August 2007.

[LW15] Arjen K. Lenstra and Benjamin Wesolowski. A random zoo: Sloth, unicorn, and trx. Technical Report 366, 2015.

[LYAX18] Kang Li, Rupeng Yang, Man Ho Au, and Qiuliang Xu. Practical Range Proof for Cryptocurrency Monero with Provable Security. In *Information and Communications Security*, 2018.

[MCEYA11] Mohammed Meziani, Pierre-Louis Cayrel, and Sidi Mohamed El Yousfi Alaoui. 2SC: An Efficient Code-Based Stream Cipher. In *Information Security and Assurance*, 2011.

[Mei12] Rebecca Meissen. *A Mathematical Approach to Fully Homomorphic Encryption*. PhD Thesis, Worcester Polytechnic Institute, 2012.

[MHC12] Mohammed Meziani, Gerhard Hoffmann, and Pierre-Louis Cayrel. Improving the Performance of the SYND Stream Cipher. In *Progress in Cryptology - AFRICACRYPT 2012*, 2012.

[MMR+17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proc. of AISTATS*, 2017.

[MMV11] Mohammad Mahmoody, Tal Moran, and Salil Vadhan. Time-Lock Puzzles in the Random Oracle Model. In *Advances in Cryptology – CRYPTO 2011*, 2011.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with Small Parameters. In *Advances in Cryptology – CRYPTO 2013*, 2013.

[MR02] Silvio Micali and Ronald L. Rivest. Micropayments Revisited. In *Topics in Cryptology — CT-RSA 2002*, 2002.

[MT09] Ravi Montenegro and Prasad Tetali. How long does it take to catch a wild kangaroo? In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, May 2009.

[MT19] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic Time-Lock Puzzles and Applications. In *CRYPTO*, 2019.

[MVR99] Silvio Micali, Salil Vadhan, and Michael Rabin. Verifiable Random Functions. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, October 1999.

[Nie02] Jesper Buus Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In *Advances in Cryptology — CRYPTO 2002*, volume 2442. 2002.

[NIS17] NIST STS. *Cryptographic Key Length Recommendation.* 2017.

[NS08] Arvind Narayanan and Vitaly Shmatikov. Robust De-anonymization of Large Sparse Datasets. In *2008 IEEE Symposium on Security and Privacy (Sp 2008)*, May 2008.

[PAH+18] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai. Privacy-Preserving Deep Learning via Additively Homomorphic Encryption. *IEEE Trans. Inf. Forensics Secur.*, 13(5), May 2018.

[Pai99] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology — EUROCRYPT '99*, 1999.

[Par18] Stuart L Pardau. The california consumer privacy act: Towards a european-style privacy regime in the united states. *J Tech Pol*, 23, 2018.

[PB10] K. Peng and F. Bao. An Efficient Range Proof Scheme. In *2010 IEEE Second International Conference on Social Computing*, August 2010.

[PBP18] Elena Pagnin, Carlo Brunetta, and Pablo Picazo-Sanchez. HIKE: Walking the Privacy Trail. In *Cryptology and Network Security*, 2018.

[Pei16] Chris Peikert. A Decade of Lattice Cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4), March 2016.

[PJ17] M. Panjwani and M. Jäntti. Data Protection Security Challenges in Digital IT Services: A Case Study. In *2017 International Conference on Computer and Applications (ICCA)*, September 2017.

[PO14] Alberto Peinado and Andrés Ortiz. Prediction of Sequences Generated by LFSR Using Back Propagation MLP. In *International Joint Conference SOCO'14-CISIS'14-ICEUTE'14*, 2014.

[Pol78] John M Pollard. Monte Carlo methods for index computation (mod p). *Math. Comput.*, 32(143), 1978.

[Pol00] J. M. Pollard. Kangaroos, Monopoly and Discrete Logarithms. *J. Cryptol.*, 13(4), September 2000.

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to Delegate and Verify in Public: Verifiable Computation from Attribute-Based Encryption. In *Theory of Cryptography*, 2012.

[PST13] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of Correct Computation. In *Theory of Cryptography*, 2013.

[PWH+17] Dimitrios Papadopoulos, Duane Wessels, Shumon Huque, Moni Naor, Jan Včelák, Leonid Reyzin, and Sharon Goldberg. *Making NSEC5 Practical for DNSSEC*. 2017.

[RAD78] R L Rivest, L Adleman, and M L Dertouzos. On Data Banks and Privacy Homomorphisms. *Found. Secure Comput. Acad. Press*, 1978.

[Reg10] Oded Regev. The Learning with Errors Problem (Invited Survey). In *Proceedings of the 2010 IEEE 25th Annual Conference on Computational Complexity*, 2010.

[RSW96] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock Puzzles and Timed-release Crypto. Technical report, Massachusetts Institute of Technology, 1996.

[SAL07] Dario L. M. Sacchi, Franca Agnoli, and Elizabeth F. Loftus. Changing history: Doctored photographs affect memory for past public events. *Appl. Cogn. Psychol.*, 21(8), December 2007.

[Sha48] C. E. Shannon. A Mathematical Theory of Communication. *Bell Syst. Tech. J.*, 27(3), 1948.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11), November 1979.

[SS15] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[SSM14] Bas Stottelaar, Jeroen Senden, and Lorena Montoya. Online social sports networks as crime facilitators. *Crime Sci*, 3(1), August 2014.

[SSV19] Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Publicly Verifiable Proofs from Blockchains. In *PKC*, 2019.

[ST96] Moshe Sipper and Marco Tomassini. Generating Parallel Random Number Generators by Cellular Programming. *Int. J. Mod. Phys. C*, 07(02), April 1996.

[ST13] W. A. R. D. Souza and A. Tomlinson. A Distinguishing Attack with a Neural Network. In *2013 IEEE 13th International Conference on Data Mining Workshops*, December 2013.

[Sta96] Markus Stadler. Publicly Verifiable Secret Sharing. In *Advances in Cryptology — EUROCRYPT '96*, 1996.

[STBK+18] Meltem Sönmez Turan, Elaine Barker, John Kelsey, Kerry McKay, Mary Baish, and Michael Boyle. Recommendation for the Entropy Sources Used for Random Bit Generation. Technical Report NIST Special Publication (SP) 800-90B, National Institute of Standards and Technology, January 2018.

[Ste89] Jacques Stern. A method for finding codewords of small weight. In *Coding Theory and Applications*, 1989.

[Ste96] J. Stern. A new paradigm for public key identification. *IEEE Trans. Inf. Theory*, 42(6), November 1996.

[Str18] Strava. Strava. https://www.strava.com, November 2018.

[SUM13] Petr Svenda, Martin Ukrop, and Vashek Matyáš. Towards cryptographic function distinguishers with evolutionary circuits. In *2013 International Conference on Security and Cryptography (SECRYPT)*, July 2013.

[SV10] N. P. Smart and F. Vercauteren. Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. In *Public Key Cryptography – PKC 2010*, 2010.

[TB19] Florian Tramèr and Dan Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In *Proceedings of ICLR*, 2019.

[THH+09] Brian Thompson, Stuart Haber, William G. Horne, Tomas Sander, and Danfeng Yao. Privacy-Preserving Computation and Verification of Aggregate Queries on Outsourced Databases. In *Privacy Enhancing Technologies*, volume 5672. 2009.

[TLM18] Georgia Tsaloli, Bei Liang, and Aikaterini Mitrokotsa. Verifiable Homomorphic Secret Sharing. In *Provable Security (ProvSec), 2018*, volume 11192, 2018.

[TM20] Georgia Tsaloli and Aikaterini Mitrokotsa. Sum it up: Verifiable additive homomorphic secret sharing. In *Information Security and Cryptology – ICISC 2019*, 2020.

[Var57] R. R. Varshamov. Estimate of the Number of Signals in Error Correcting Codes. *Docklady Akad Nauk SSSR*, 117, 1957.

[vGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully Homomorphic Encryption over the Integers. In *Advances in Cryptology – EUROCRYPT 2010*, 2010.

[Wes19] Benjamin Wesolowski. Efficient Verifiable Delay Functions. In *EUROCRYPT*, 2019.

[WF02] Ian H. Witten and Eibe Frank. Data mining: Practical machine learning tools and techniques with Java implementations. *SIGMOD Rec.*, 31(1), March 2002.

[WS19] Joanne Woodage and Dan Shumow. An Analysis of NIST SP 800-90A. In *Advances in Cryptology – EUROCRYPT 2019*, 2019.

[XEQ18] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*, 2018.

[XLL+20] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin. VerifyNet: Secure and Verifiable Federated Learning. *IEEE Trans. Inf. Forensics Secur.*, 15, 2020.

[YS16] Yu Yu and John Steinberger. Pseudorandom Functions in Almost Constant Depth from Low-Noise LPN. In *Advances in Cryptology – EUROCRYPT 2016*, volume 9666. 2016.

[ZZL18] Zhicheng Zhao, Yaqun Zhao, and Fengmei Liu. The Research of Cryptosystem Recognition Based on Randomness Test's Return Value. In *Cloud Computing and Security*, 2018.