Code-Based Zero Knowledge PRF Arguments

Carlo Brunetta, Bei Liang, and Aikaterini Mitrokotsa

Chalmers University of Technology, Gothenburg, Sweden
{brunetta,lbei,aikmitr}@chalmers.se

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.* [19] has recently addressed this problem based on lattice-based assumptions and more precisely, based on the *Learning-With-Rounding* (LWR) problem [4] 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 [5], 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 [5], 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 $\boldsymbol{H} \in \mathbb{F}_2^{r \times n}$ and a uniformly random syndrome $\boldsymbol{y} \in \mathbb{F}_2^r$, find a vector (word) $\boldsymbol{x} \in \mathbb{F}_2^n$ with Hamming weight ω, such that $\boldsymbol{H} \cdot \boldsymbol{x}^\intercal = \boldsymbol{y}^\intercal$. Berlekamp, McEliece and Tilborg [5] 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 [9,23]) it is the hardest to solve, when the weights of the words (*i.e.,* $\boldsymbol{x} \in \mathbb{F}_2^n$) are in the neighbourhood of the Gilbert-Varshamov bound [14,24]. 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) [3], 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 [13] used regular words to improve Fischer and Stern's code-based PRG [12]. 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 [12] 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 $\boldsymbol{H} \in \mathbb{F}_2^{\lfloor \rho n \rfloor \times n}$, $\boldsymbol{x} \in \mathbb{F}_2^n$, $\omega = \lfloor \delta n \rfloor$ 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 $G_{\rho,\delta}(x) = \boldsymbol{H} \cdot \boldsymbol{x}^\intercal$ 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 [12] provided an iterative generator, which after computing $\boldsymbol{y} = \boldsymbol{H} \cdot \boldsymbol{x}^\intercal$, separates y as $\boldsymbol{y} = \boldsymbol{y}_1 \| \boldsymbol{y}_2$, where $\boldsymbol{y}_1$ denotes the first $\log_2 \binom{n}{\delta n}$ bits of $\boldsymbol{y}$ and $\boldsymbol{y}_2$ denotes the remaining bits. It outputs $\boldsymbol{y}_2$ and uses $\boldsymbol{y}_1$ 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 [16] 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 [12] into the sequential GGM paradigm [16], 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 [12] 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 [22] based on the hardness of the SD problem and its variants [8,2], as well as identity-based identification

schemes [7,10]. There have also been proposed ZK proofs of plaintext knowledge based on the McEliece and the Niederreiter cryptosystems [17], 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 [11]. 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 [3].

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 $\boldsymbol{y}$ is indeed correctly computed from the code-based PRF with a secret key $\boldsymbol{k}$ held by the prover on the input $\boldsymbol{x}$. 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 [12,13,21] or other code-based assumptions [25], 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 $\boldsymbol{H}_0, \boldsymbol{H}_1 \in \mathbb{F}_2^{r\times n}$ where $r = w \cdot b$ and $n = w \cdot 2^b$ and $f : \mathbb{F}_2^r \to \mathbb{F}_2^{2r}$ as:

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

For an input bit string $\boldsymbol{x} \in \mathbb{F}_2^t$ 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:

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

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 [21].

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 $\boldsymbol{y}$ is correctly computed from the PRF using a secret key $\boldsymbol{k}$ on input $\boldsymbol{x}$, namely $\boldsymbol{y} = f_{x_t}(f_{x_{t-1}}(\cdots(f_{x_1}(\boldsymbol{k}))\cdots)$. Without loss of generality, let us consider the case for input length of $t = 2$. Given $\boldsymbol{x} = (x_1, x_2)$, according to our PRF construction, it holds:

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

If we reveal all the intermediate results, *i.e.,* the value $\boldsymbol{y}^1 = f_{x_1}(\boldsymbol{k})$ which is exactly the seed used to compute the next GGM iteration *i.e.,* the value $\boldsymbol{y} = \boldsymbol{y}^2 = f_{x_2}(\boldsymbol{y}^1)$, then it is possible for a malicious verifier to compute the PRF on different inputs $\boldsymbol{x}' = (x_1, 1 - x_2)$ (without knowing the secret key $\boldsymbol{k}$), 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:

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

By embedding the above technique into Stern's ZK protocol framework [22], we obtain an interactive ZK argument system, in which, given the input and output values $\boldsymbol{x}, \boldsymbol{y}$,the prover is able to prove that $\boldsymbol{y} = \mathsf{PRF}_{\boldsymbol{k}}(\boldsymbol{x})$ is indeed the evaluation of $f_{x_t}(f_{x_{t-1}}(\cdots(f_{x_1}(\boldsymbol{k}))\cdots)$. The protocol is repeated many times to achieve negligible soundness error.

Related Work. Libert *et al.* [19] 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 [22] based on lattices. Brunetta *et al.* [6] 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 [16] 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 [12] 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 [13], which is an improvement of Fischer-Stern's PRG, revisited as a stream-cipher. Meziani *et al.* proposed 2SC [20], a code-based sponge-function stream cipher. Shortly after, Meziani *et al.* improved the SYND cipher and defined X-SYND [21], 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 [22] published a code-based identification protocol with a zero-knowledge property. Different improved versions are defined by Aguilar *et al.* [2] or Cayrel *et al.* [8] in order to reduce the soundness error. In our constructions, we have employed the original zero-knowledge identification protocol proposed by Stern [22] , 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 [22].

Let $\mathbb{N}$ be the set of positive integers and let the uniform sampling of x in a set X defined as $x \in_{\$} X$. 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 $\boldsymbol{x} \in \mathbb{F}_2^b$. We denote with $\mathsf{B2I}(\boldsymbol{x})$ the map that takes a binary string $\boldsymbol{x}$ 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 $\boldsymbol{y} \in \mathbb{F}_q^n$ are called *words* and, if they are part of the code, *i.e.* $\boldsymbol{y} \in \mathcal{C}$, then, they are called *codewords*. The weight of a word $\boldsymbol{x}$ is denoted as $\mathsf{wt}(\boldsymbol{x})$ and it counts the number of non-zero components of the word $\boldsymbol{x}$. A code $\mathcal{C}$ can be represented by a generator matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$ as $\mathcal{C} = \{\boldsymbol{x} \cdot \boldsymbol{G} \mid \boldsymbol{x} \in \mathbb{F}_q^k\}$, 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 $\boldsymbol{H} \in \mathbb{F}_q^{(n-k) \times n}$. For the matrix $\boldsymbol{H}$, it holds

$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{F}_q^n \mid \boldsymbol{H} \cdot \boldsymbol{x}^\intercal = 0\}$. 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 $\boldsymbol{H} \in \mathbb{F}_2^{r \times n}$ of the code $\mathcal{C}$ of dimension k over $\mathbb{F}_2^n$, in which we consider $r = n - k$.

Assumption 1 (Binary Syndrome Decoding (SD)) *Given a binary matrix $\boldsymbol{H} \in \mathbb{F}_2^{r \times n}$, a binary vector $\boldsymbol{y} \in \mathbb{F}_2^r$ and an integer $w > 0$, find a* ***word*** *$\boldsymbol{x} \in \mathbb{F}_2^n$ such that $\mathsf{wt}(\boldsymbol{x}) = w$ and $\boldsymbol{H} \cdot \boldsymbol{x}^\intercal = \boldsymbol{y}$.*

The SD problem is known to be NP-complete [5]. We are interested in a simplified version of the SD problem in which the word $\boldsymbol{x}$ is **regular**, *i.e.* for $\boldsymbol{x}$ 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 2 (Regular Syndrome Decoding (RSD(n, r, w))) *Given a binary matrix $\boldsymbol{H} \in \mathbb{F}_2^{r \times n}$, a binary vector $\boldsymbol{y} \in \mathbb{F}_2^r$ and an integer $w > 0$, find a* ***regular word*** *$\boldsymbol{x} \in \mathbb{F}_2^n$ such that $\mathsf{wt}(\boldsymbol{x}) = w$ and $\boldsymbol{H} \cdot \boldsymbol{x}^\intercal = \boldsymbol{y}$.*

Augot *et al.* [3] prove the NP-completeness of the RSD problem and we will base the security of our constructions on this specific problem.

Stern's protocol [22] is a zero-knowledge sigma-protocol that describes the language L defined as the elements $(\boldsymbol{M}, \boldsymbol{y}) \in \mathbb{F}_2^{r \times n} \times \mathbb{F}_2^r$ of which there exists a witness $\boldsymbol{s} \in \mathbb{F}_2^n$, such that $\mathsf{wt}(\boldsymbol{s}) = w$ and $\boldsymbol{M} \cdot \boldsymbol{s} = \boldsymbol{y}$. 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 $\boldsymbol{s}$ given the statement $(\boldsymbol{M}, \boldsymbol{y})$.

Theorem 1 (Stern's protocol). *From the original paper [22], Stern's protocol, as reported in Figure 1, is correct, has soundness probability of $\frac{2}{3}$ and it is zero-knowledge. Let π a permutation of the set $\{1, \ldots, n\}$ if we assume that $|\pi| > n$, and $|\mathsf{Com}|$ is the commitment length, then the communication cost of the protocol is:*

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

3 Code-Based PRF

In this section, inspired by Gaborit's [13] and Meziani's [21] code-based stream ciphers, we define our own simple PRG f 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 [16].

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

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

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

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* $c = 0$, $(\pi(\boldsymbol{r}), \pi(\boldsymbol{s}))$ and ρ_1, ρ_2; *if* $c = 1$, $(\boldsymbol{r} \oplus \boldsymbol{s}, \pi)$ and ρ_0, ρ_2; and *if* $c = 2$, $(\boldsymbol{r}, \pi)$ and ρ_0, ρ_1.

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

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

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

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

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

Figure 1. Stern's protocol description.

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 2 is hard. Consider the binary words $\boldsymbol{s} \in \mathbb{F}_2^n$ of length n and composed by w blocks of length 2^b, *i.e.* $\boldsymbol{s} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_w)$ such that every block $\boldsymbol{s}_j$ has weigth $\mathsf{wt}(\boldsymbol{s}_j) = 1$. 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 $\boldsymbol{y} \in \mathbb{F}_2^r$ and outputs a regular word $\boldsymbol{s} \in \mathbb{F}_2^n$ such that $\mathsf{wt}(\boldsymbol{s}) = w$ and that is computed as follows.

Firstly, the binary string $\boldsymbol{y}$ is divided into w blocks as $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_w)$ of which each block $\boldsymbol{y}_i$ is a binary string with length b. Then, for every $j \in \{1, \ldots, t\}$, we compute the integer value n_j represented by the block $\boldsymbol{y}_j$ and denote it as $\mathsf{B2I}(\boldsymbol{y}_j) = n_j$. In this way, we transform the vector $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_w)$ into a vector of integers $(n_1, \cdots, n_w)$, where every n_j is contained in the interval $\{0, \ldots, 2^b - 1\}$. 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 $\boldsymbol{e}_{n_j+1} \in \mathbb{F}_2^{2^b}$, 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 $(\boldsymbol{e}_{n_1+1}, \cdots, \boldsymbol{e}_{n_w+1})$ that are concatenated and output by ϕ. In summary, the map ϕ is computed as:

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

It is trivial to observe that $\boldsymbol{s}$ is a regular word of length n and weigth w since $\boldsymbol{s}$ is the concatenation of w canonical vectors of length 2^b and the weight $\mathsf{wt}(\boldsymbol{s})$ 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 $\boldsymbol{y} = (01\|11\|00)$ would be transformed into the regular word $\phi(\boldsymbol{y}) = \boldsymbol{s} = (\boldsymbol{e}_2\|\boldsymbol{e}_4\|\boldsymbol{e}_1) = (0100\|0001\|1000)$.

After defining the map ϕ, we are interested in developing a pseudorandom generator (PRG) based on the RSD assumption (see Assumption 2), inspired by Meziani's [21] stream-cipher design. Let us first report both definitions.

Definition 1 (Pseudorandom Generator (PRG) [18]). *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 [12], 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 [21] code-based stream cipher, depicted in Figure 2.

Definition 2 (X-SYND Stream Cipher [21]). *Let $w, b \in \mathbb{N}$ be positive integers and define $n = w2^b$, $r = wb$. Let $\boldsymbol{A}_0, \boldsymbol{A}_1 \in_{\$} \mathbb{F}_2^{r\times n}$ be random binary matrices. Define the **X-SYND stream cipher** as:*

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

Similarly to X-SYND, let $\boldsymbol{A}_0, \boldsymbol{A}_1 \in \mathbb{F}_2^{r\times n}$, 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:

$$f(\boldsymbol{k}) = \begin{pmatrix} \boldsymbol{A}_0 \\ \boldsymbol{A}_1 \end{pmatrix} \cdot \phi(\boldsymbol{k})^\intercal = \begin{pmatrix} \boldsymbol{A}_0 \cdot \phi(\boldsymbol{k})^\intercal \\ \boldsymbol{A}_1 \cdot \phi(\boldsymbol{k})^\intercal \end{pmatrix} = \begin{pmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \end{pmatrix} \tag{1}$$

It has to be observed that f is indeed Meziani *et al.*'s [21] 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.

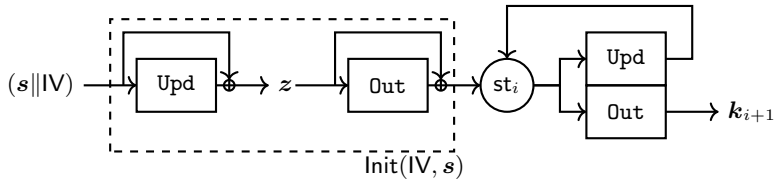

Figure 2. A high-level representation of the X-SYND stream cipher.

Proposition 1. *f is a PRG that reduce to a $\mathsf{RSD}(n, 2r, w)$ problem (Assum. 2).*

Proof. We sketch the main idea of the proof in two parts, that follow the same reasoning as Meziani *et al.*'s [21] 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 $\boldsymbol{x}$ is equivalent of knowing the vector $\boldsymbol{k}$ such that $\phi(\boldsymbol{k}) = \boldsymbol{x}$.

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

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

The second step is pseudorandomness and to prove it, we use the fact that Meziani *et al.* prove in Theorem 2 [21], using Goldreich-Levin hard-core bit theorem [15], 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 [16]. For the sake of clarity, let us report the PRF definition and the GGM construction.

Definition 3 (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 4 (GGM Construction [16]). *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 $\boldsymbol{x} \in \{0,1\}^m$, we define the **GGM pseudorandom function** $F_s : \{0,1\}^m \to \{0,1\}^n$ as*

$$F_s(\boldsymbol{x}) = F_s\big((x_1, \ldots, x_\ell)\big) = G_{x_\ell}(G_{x_{\ell-1}}(\cdots(G_{x_1}(s))\cdots) \tag{2}$$

Theorem 2. *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. (1) with the GGM construction of Eq. (2). For readability, we will always denote the key in subscript, *i.e.* $\mathsf{PRF}(\boldsymbol{k}, \boldsymbol{x}) = \mathsf{PRF}_{\boldsymbol{k}}(\boldsymbol{x})$. Formally we have,

$$\mathsf{PRF}_{\boldsymbol{k}}(\boldsymbol{x}) = \mathsf{PRF}_{\boldsymbol{k}}\big((x_1, \cdots, x_t)\big) = f_{x_t}\Big(f_{x_{t-1}}\Big(\cdots\big(f_{x_1}(\boldsymbol{k})\big)\Big)\cdots\Big) \tag{3}$$

Corollary 1. *By Theorem 2,* 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 [22] 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 $\big((\boldsymbol{M}, \boldsymbol{y}), \boldsymbol{s}\big)$.

Briefly, Stern's protocol allows a prover $\mathcal{P}$ to prove the knowledge of a witness $\boldsymbol{s}$ with weight w to a verifier $\mathcal{V}$, that holds a public statement $\big(\boldsymbol{M}, \boldsymbol{y}\big)$. The statement and the witness are related to the equation $\boldsymbol{M} \cdot \boldsymbol{s}^\intercal = \boldsymbol{y}^\intercal$.

At a first glance, we can observe that our PRG f 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 $\boldsymbol{y}^i$ 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 $\boldsymbol{A}_{x_{(i+1)}} \cdot \phi(\boldsymbol{y}^i)^\intercal = \boldsymbol{y}^{(i+1)\intercal}$. Since ϕ has no-linear representation, it is indeed impossible to re-write the equation as a single matrix $\overline{\boldsymbol{M}} \in \mathbb{F}_2^{r \times n}$ that multiplies only the secret initial vector $\phi(\boldsymbol{k})$.

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 $\phi(\boldsymbol{k})$. 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 $\boldsymbol{k} = \boldsymbol{y}^0$ and $\boldsymbol{x} = (x_1, \ldots, x_t)$ and $\boldsymbol{y}^t = \mathsf{PRF}(\boldsymbol{k}, \boldsymbol{x})$. Then, it formally holds that:

$$\begin{cases} \boldsymbol{A}_{x_1} \cdot \phi(\boldsymbol{y}^0)^\intercal = \boldsymbol{y}^{1\intercal} \\ \boldsymbol{A}_{x_2} \cdot \phi(\boldsymbol{y}^1)^\intercal = \boldsymbol{y}^{2\intercal} \\ \quad \cdots \\ \boldsymbol{A}_{x_t} \cdot \phi(\boldsymbol{y}^{t-1})^\intercal = \boldsymbol{y}^{t\intercal} \end{cases} \iff \begin{pmatrix} \boldsymbol{A}_{x_1} & 0 & & \\ 0 & \boldsymbol{A}_{x_2} & 0 & \\ & \ddots & \ddots & \ddots \\ & & 0 & \boldsymbol{A}_{x_t} \end{pmatrix} \cdot \begin{pmatrix} \phi(\boldsymbol{y}^0)^\intercal \\ \phi(\boldsymbol{y}^1)^\intercal \\ \vdots \\ \phi(\boldsymbol{y}^{t-1})^\intercal \end{pmatrix} = \begin{pmatrix} \boldsymbol{y}^{1\intercal} \\ \boldsymbol{y}^{2\intercal} \\ \vdots \\ \boldsymbol{y}^{t\intercal} \end{pmatrix} \tag{4}$$

Unfortunately, this representation has a security-flaw that allows a malicious adversary $\mathcal{A}$ to compute the PRF on different inputs $\boldsymbol{x}'$ without requiring the knowledge of $\boldsymbol{k}$. 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 $\boldsymbol{x}$, $\mathcal{A}$ can either ask to obtain just the value $\mathsf{PRF}(\boldsymbol{k}, \boldsymbol{x})$ or to obtain the transcript of the execution of Stern's protocol, which contains $\mathsf{PRF}(\boldsymbol{k}, \boldsymbol{x})$ 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 (4) 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 $\boldsymbol{y}^{t-1}$ and correctly compute $\boldsymbol{A}_{1-x_t} \cdot \phi(\boldsymbol{y}^{t-1})^{\mathsf{T}} = \mathsf{PRF}\big(\boldsymbol{k}, \boldsymbol{x}'\big)$, which is a valid PRF evaluation of the input $\boldsymbol{x}'$ with a different t-th component. With this knowledge, $\mathcal{A}$ can query the challenger on $\boldsymbol{x}'$ 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 $\boldsymbol{x}'$ except the ones that have a different first input-bit x_1. This is because $\mathcal{A}$ does not hold the pre-computation $\boldsymbol{y}^0$, which is exactly the secret key $\boldsymbol{k}$.

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 2 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 $\boldsymbol{w}$ 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 $\boldsymbol{w}$ be a regular word of length $n = w \cdot 2^b$ that representa $\boldsymbol{w}$ as the concatenation of w canonical vectors, *i.e.* $\boldsymbol{w} = \big(\boldsymbol{e}_{n_1} \mid \cdots \mid \boldsymbol{e}_{n_w}\big)$. 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 $\boldsymbol{\psi}$ as:

$$\boldsymbol{\psi} = \left(\underbrace{\left(\mathsf{I2B}_b(0)^\intercal \| \dots \| \mathsf{I2B}_b\left(2^b-1\right)^\intercal\right) \| \ \cdots \ \| \left(\mathsf{I2B}_b(0)^\intercal \| \dots \| \mathsf{I2B}_b\left(2^b-1\right)^\intercal\right)}_{w \text{ times}} \right) \tag{5}$$

By notation abuse, let the evaluation of the map ψ be the matrix multiplication with the matrix $\boldsymbol{\psi}$ in Equation (5). Formally,

$$\psi(\boldsymbol{w}) = \boldsymbol{\psi} \cdot \boldsymbol{w}^\intercal = \boldsymbol{\psi} \cdot \left(\boldsymbol{e}_{n_1+1} \mid \cdots \mid \boldsymbol{e}_{n_w+1}\right)^\intercal = \left(\mathsf{I2B}_b(n_1) \| \dots \| \mathsf{I2B}_b(n_t)\right)^\intercal$$

Lemma 1. *For all $\boldsymbol{y} \in \mathbb{F}_2^r$, it holds $(\psi \circ \phi)(\boldsymbol{y}) = \boldsymbol{y}$,* i.e., *$\psi$ is the inverse of ϕ.*

Proof. Ad oculos, let $\boldsymbol{y} = (\boldsymbol{y}_1 \| \dots \| \boldsymbol{y}_w)$.

$$\begin{aligned}\big(\psi \circ \phi\big)(\boldsymbol{y}) &= \psi\Big(\left(\boldsymbol{e}_{\mathsf{B2I}(\boldsymbol{y}_1)+1} \| \dots \| \boldsymbol{e}_{\mathsf{B2I}(\boldsymbol{y}_w)+1}\right) \Big) \\ &= \Big(\big(\mathsf{I2B}_b \circ \mathsf{B2I}\big)(\boldsymbol{y}_1) \| \dots \| \big(\mathsf{I2B}_b \circ \mathsf{B2I}\big)(\boldsymbol{y}_w) \Big) = (\boldsymbol{y}_1 \| \dots \| \boldsymbol{y}_w) = \boldsymbol{y} \quad \square\end{aligned}$$

Given the invertibility property, we are now able to further modify and fix the naïve approach presented in Eq. (4). For every $j \in \{1, \dots, (t-1)\}$, let us rewrite the equation by moving all the addends to the left-hand side. Formally,

$$\begin{aligned}\boldsymbol{A}_{x_i} \cdot \phi(\boldsymbol{y}^{i-1})^\intercal = \boldsymbol{y}^{i\intercal} &\iff \boldsymbol{A}_{x_i} \cdot \phi(\boldsymbol{y}^{i-1})^\intercal \oplus \boldsymbol{y}^{i\intercal} = \boldsymbol{0} \\ &\iff \boldsymbol{A}_{x_i} \cdot \phi(\boldsymbol{y}^{i-1})^\intercal \oplus \big(\psi \circ \phi\big)^\intercal(\boldsymbol{y}^i) = \boldsymbol{0} \\ &\iff \boldsymbol{A}_{x_i} \cdot \phi(\boldsymbol{y}^{i-1})^\intercal \oplus \boldsymbol{\psi} \cdot \phi(\boldsymbol{y}^i)^\intercal = \boldsymbol{0}\end{aligned} \tag{6}$$

By rewriting Eq. (6) in Eq. (4), define $\widehat{\boldsymbol{M}} \in \mathbb{F}_2^{tr \times tn}$ and $\widehat{\boldsymbol{s}} \in \mathbb{F}_2^{tn}, \widehat{\boldsymbol{y}} \in \mathbb{F}_2^{tr}$ as:

$$\widehat{\boldsymbol{M}} \cdot \widehat{\boldsymbol{s}} := \begin{pmatrix} \boldsymbol{A}_{x_1} & \boldsymbol{\psi} & & \\ & \ddots & \ddots & \\ & & \boldsymbol{A}_{x_{t-1}} & \boldsymbol{\psi} \\ & & & \boldsymbol{A}_{x_t} \end{pmatrix} \cdot \begin{pmatrix} \phi(\boldsymbol{y}^0)^\intercal \\ \vdots \\ \phi(\boldsymbol{y}^{t-2})^\intercal \\ \phi(\boldsymbol{y}^{t-1})^\intercal \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \\ \boldsymbol{y}^{t\intercal} \end{pmatrix} =: \widehat{\boldsymbol{y}} \tag{7}$$

Proposition 2. *Let $\widehat{\boldsymbol{M}}, \widehat{\boldsymbol{s}}, \widehat{\boldsymbol{y}}$ as defined in Eq. (7). The related Stern language,*

$$\hat{L} = \left\{ \left(\widehat{\boldsymbol{M}}, \widehat{\boldsymbol{y}}\right) \;\middle|\; \exists\, \widehat{\boldsymbol{s}} : \ \mathsf{wt}(\widehat{\boldsymbol{s}}) = wt \ \wedge \ \widehat{\boldsymbol{M}} \cdot \widehat{\boldsymbol{s}} = \widehat{\boldsymbol{y}} \right\}$$

is equivalent to the PRF evaluation language for PRF *of Eq. (3),* i.e.,

$$L_{\mathsf{PRF}} = \{(\boldsymbol{x}, \boldsymbol{y}) \mid \exists\, \boldsymbol{k} : \ \mathsf{PRF}(\boldsymbol{k}, \boldsymbol{x}) = \boldsymbol{y}\}$$

Proof. Since the global parameters n, r, w are known, the matrix $\boldsymbol{\psi}$ is defined and it is trivial to observe that $\widehat{\boldsymbol{M}}$ can be reconstructed with the knowledge of the matrices $\boldsymbol{A}_0, \boldsymbol{A}_1$ and $\boldsymbol{x}$. Furthermore, since $t-1$ components of $\widehat{\boldsymbol{y}}$ are zero,

only $\boldsymbol{y}^t$ is needed to correctly reconstruct the language statement's vector. To this point, we can rewrite $\hat{L}$ as:

$$\hat{L} = \left\{ \left((\boldsymbol{A}_0, \boldsymbol{A}_1, \boldsymbol{x}), \boldsymbol{y}^t \right) \;\middle|\; \exists\, \widehat{\boldsymbol{s}} :\ \mathsf{wt}(\widehat{\boldsymbol{s}}) = wt \ \wedge\ \widehat{\boldsymbol{M}} \cdot \widehat{\boldsymbol{s}} = \widehat{\boldsymbol{y}} \right\}$$

Since the PRF is defined by the matrices $\boldsymbol{A}_0, \boldsymbol{A}_1$, $\boldsymbol{y}^t = \mathsf{PRF}(\boldsymbol{k}, \boldsymbol{x})$ and the matrix multiplication represents the GGM iterated PRF computations, we have

$$\hat{L} = L'_{\mathsf{PRF}} = \{(\boldsymbol{x}, \boldsymbol{y}) \mid \exists\, \widehat{\boldsymbol{s}} :\ \mathsf{wt}(\widehat{\boldsymbol{s}}) = wt \ \wedge\ \mathsf{PRF}(\boldsymbol{k}, \boldsymbol{x}) = \boldsymbol{y}\}$$

and we are left to prove that possessing the PRF secret key $\boldsymbol{k} \in \mathbb{F}_2^r$ is equivalent to knowing all the regular-words and partial evaluations $\boldsymbol{y}^j \in \mathbb{F}_2^n$ used in the GGM transformation, for all indexes $j \in \{0, \ldots, (t-1)\}$.

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 $\boldsymbol{k} = \boldsymbol{y}^0$ allows the computation of all the other partial evaluations $\boldsymbol{y}^j$ for $j \in \{1, \ldots, (t-1)\}$ 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 1 and Proposition 2, executing Stern's protocol on* $\left(\widehat{\boldsymbol{M}}, \widehat{\boldsymbol{s}}, \widehat{\boldsymbol{y}}\right)$ *as defined in Eq. (7), 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 [1] 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 3. Let us explain the scenario in detail, while employing our ZK protocol.

The clients prepare a document $\boldsymbol{x}$ containing all the info that they are willing to publish. The notary, in possess of a secret key $\boldsymbol{k}$, will verify the document's validity and he/she will publicly commit to the document with $\boldsymbol{y} = \mathsf{PRF}_{\boldsymbol{k}}(\boldsymbol{x})$. A verifier $\mathcal{V}$ will be able to verify the correctness of $(\boldsymbol{x}, \boldsymbol{y})$ 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.

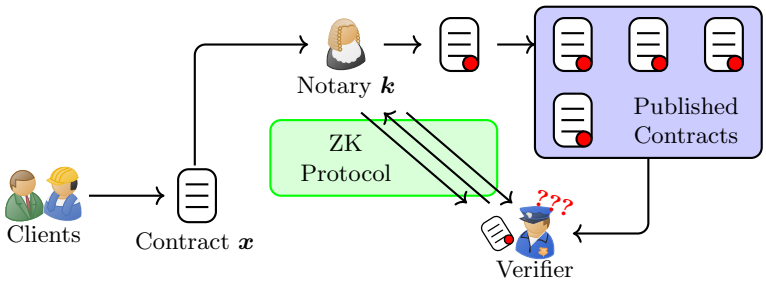

Figure 3. The whistle-blower notary problem.

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 [22] and Meziani *et al.* [21].

First of all, we overestimate the length of a permutation π of the set $\{1, \ldots, n\}$ as $|\pi| = n \log_2(n)$, which is the bit-representation of the permutation's image *w.r.t.* a fixed order, *e.g.*, $\pi = \big(\mathsf{I2B}_n(\pi(1)) \| \ldots \| \mathsf{I2B}_n(\pi(n))\big)$. By employing the *random hashing technique*, as denoted by Stern, we may use a hash function Hash 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 $|\mathsf{Hash}|$.

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:

$$\begin{aligned}\mathsf{Cost}(w,b,t,d) &= d \cdot \mathsf{Cost}_{\mathsf{Stern}}(tw2^b, twb) \\ &\leq d \cdot \Big(3|\mathsf{Hash}| + tw \cdot \Big(2^{b+1}\big(1+b+\log_2(tw)\big)+b\Big)+2\Big) \text{ bits}\end{aligned}$$

From the X-SYND definition [21], 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 $|\mathsf{Hash}| = 128$ 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 $|\boldsymbol{A}_0| + |\boldsymbol{A}_0| = 2 \cdot w2^b \cdot wb$ 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 4.

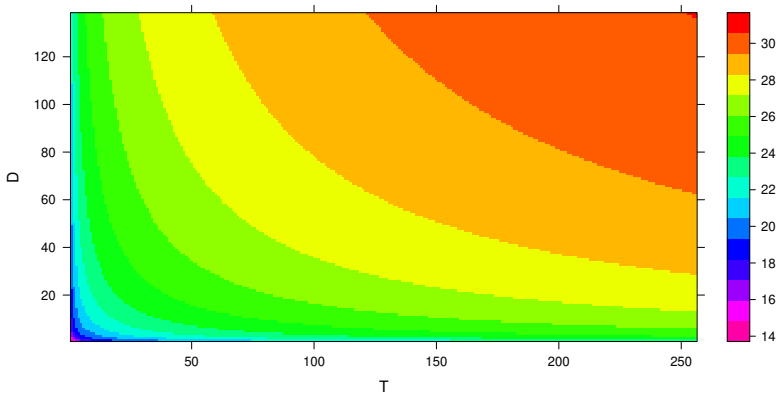

Figure 4. A heatmap plot of $\log_2\left(\frac{\mathsf{Cost}(32,8,t,d)}{8}\right)$ in which, for every t and d, we represent the communication cost in base-2 logarithmic scale. This means that a value of 20 represents 2^{20} bytes, which is 1 Megabyte.

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 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 [6] 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 [2] or Cayrel *et al.*'s [8] 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).

References

1. Adleman, L.M.: Implementing an Electronic Notary Public. In: Advances in Cryptology (1983)

2. Aguilar, C., Gaborit, P., Schrek, J.: A New Zero-Knowledge Code Based Identification Scheme with Reduced Communication. In: 2011 IEEE Information Theory Workshop. https://doi.org/10.1109/ITW.2011.6089577
3. Augot, D., Finiasz, M., Sendrier, N.: A Family of Fast Syndrome Based Cryptographic Hash Functions. In: Progress in Cryptology – Mycrypt 2005. LNCS (2005)
4. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom Functions and Lattices. In: Advances in Cryptology - EUROCRYPT 2012 (2012)
5. Berlekamp, E., McEliece, R., Van Tilborg, H.: On the Inherent Intractability of Certain Coding Problems (Corresp.). IEEE Transactions on Information Theory **24**(3) (1978)
6. Brunetta, C., Liang, B., Mitrokotsa, A.: Lattice-Based Simulatable VRFs: Challenges and Future Directions. Journal of Internet Services and Information Security (JISIS) **8**(4) (Nov 2018)
7. Cayrel, P.L., Gaborit, P., Girault, M.: Identity-Based Identification and Signature Schemes Using Correcting Codes. In: WCC. vol. 2007 (2007)
8. Cayrel, P.L., Véron, P., El Yousfi Alaoui, S.M.: A Zero-Knowledge Identification Scheme Based on the q-Ary Syndrome Decoding Problem. In: Selected Areas in Cryptography (2011)
9. Chabaud, F.: On the security of some cryptosystems based on error-correcting codes. In: Advances in Cryptology — EUROCRYPT'94
10. El Yousfi Alaoui, S.M., Cayrel, P.L., Mohammed, M.: Improved Identity-Based Identification and Signature Schemes Using Quasi-Dyadic Goppa Codes. In: Information Security and Assurance (2011)
11. Ezerman, M.F., Lee, H.T., Ling, S., Nguyen, K., Wang, H.: A Provably Secure Group Signature Scheme from Code-Based Assumptions. In: ASIACRYPT 2015
12. Fischer, J.B., Stern, J.: An Efficient Pseudo-Random Generator Provably As Secure As Syndrome Decoding. EUROCRYPT'96 (1996)
13. Gaborit, P., Lauradoux, C., Sendrier, N.: SYND: A Fast Code-Based Stream Cipher with a Security Reduction. In: 2007 IEEE International Symposium on Information Theory (Jun 2007). https://doi.org/10.1109/ISIT.2007.4557224
14. Gilbert, E.N.: A Comparison of Signalling Alphabets. The Bell System Technical Journal **31**(3) (May 1952). https://doi.org/10.1002/j.1538-7305.1952.tb01393.x
15. Goldreich, O., Levin, L.A.: A Hard-Core Predicate for All One-Way Functions. STOC '89, ACM (1989). https://doi.org/10.1145/73007.73010
16. Goldreich, O., Goldwasser, S., Micali, S.: How to Construct Random Functions. J. ACM **33**(4) (Aug 1986). https://doi.org/10.1145/6490.6503
17. Hu, R., Morozov, K., Takagi, T.: Proof of Plaintext Knowledge for Code-Based Public-Key Encryption Revisited. ASIA CCS '13 (2013). https://doi.org/10.1145/2484313.2484385
18. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. CRC press (2014)
19. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-Knowledge Arguments for Lattice-Based PRFs and Applications to E-Cash. In: Asiacrypt 2017. LNCS (Dec 2017)
20. Meziani, M., Cayrel, P.L., El Yousfi Alaoui, S.M.: 2SC: An Efficient Code-Based Stream Cipher. In: Information Security and Assurance, vol. 200 (2011). https://doi.org/10.1007/978-3-642-23141-4_11
21. Meziani, M., Hoffmann, G., Cayrel, P.L.: Improving the Performance of the SYND Stream Cipher. In: Progress in Cryptology - AFRICACRYPT 2012. LNCS (2012)
22. Stern, J.: A New Paradigm for Public Key Identification. IEEE Transactions on Information Theory **42**(6) (Nov 1996). https://doi.org/10.1109/18.556672
23. Stern, J.: A method for finding codewords of small weight. In: Coding Theory and Applications (1989)

24. Varshamov, R.R.: Estimate of the Number of Signals in Error Correcting Codes. Docklady Akad. Nauk, S.S.S.R. **117** (1957)
25. Yu, Y., Steinberger, J.: Pseudorandom Functions in Almost Constant Depth from Low-Noise LPN. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016, vol. 9666. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_6