ABSTRACT

Subliminal signatures were introduced by Simmons (CRYPTO 1983), who pointed out that a malicious signer can embed secret information into a signature. Simmons also gave an interactive protocol for subliminal-free signatures based on the discrete logarithm problem.

We propose two protocols for subliminal-free signatures, and these are the first constructions to achieve subliminal-free channels in the post-quantum setting. The core of our protocols is a verifiable random secrets (VRS) scheme, which is of independent interest. Combining the VRS with a signature scheme gives us subliminal-free signatures.

Our first construction is a scheme combining a lattice-based VRS with lattice-based signatures. The VRS uses the commitment scheme from Baum et al. (SCN 2018) and the zero-knowledge proof of shuffle of known content by Silde et al. (IN SUBMISSION) to generate verifiable randomness. The VRS can be combined with the lattice-based signature framework by Lyubashevsky (EUROCRYPT 2012) to achieve subliminal-free signatures. The concrete instantiation can be made non-interactive and it takes ≈ 10 seconds to create a subliminal-free signature of total size ≈ 13 MB.

Our second VRS construction is inspired by the "cut-and-choose" techniques used by Katz et al. (CCS 2018) and Beullens (EUROCRYPT 2020). It is based purely on one-way functions, and can be combined with any "hash-then-sign" signature scheme. Our most practical instantiation only takes 1 s to generate a subliminal-free signature of size 3 KB, where a malicious signer has probability 2^{-10} to embed subliminal information into the signature.

Subliminal digital signatures can e.g. be a threat against two-factor authentication systems when the second device is malicious. Boneh et al. (IEEE S&P 2019) gave a solution to this problem for signatures based on the hardness of computing discrete logarithms over elliptic curves. Our protocol can be an alternative solution for lattice-based signatures.

ACKNOWLEDGMENTS

You are now reading my master thesis in cryptography, which is a part of my integrated Ph.D.-program in mathematical sciences at NTNU. I'm very grateful to my supervisor Kristian Gjøsteen, our discussions have been the most enlightening moments of my academic career. I also want to thank Herman Galteland, in which parts of this thesis was co-authored. This also concludes all coursework and teaching in my Ph.D., and although interesting and valuable, I'm excited to focus purely on research going forward.

Thanks to my collaborators Diego Aranha, Carsten Baum, Thor Tunge and Kristian for fruitful work on lattice-based zero-knowledge protocols, in which this thesis builds upon. Especially thanks to Carsten for hosting me for one week in Aarhus November 2019, and for inviting me back for the full upcoming fall semester. I would also like to thank Colin Boyd, Bor de Kock and Thomas Haines for very exciting ongoing collaboration within the area of lattice-based cryptography.

I'm very grateful to be a member of the NTNU Applied Cryptology Lab, and I consider you both colleagues and friends. I've also had the honor of teaching linear algebra the past three semesters together with Aslak Buan, Gereon Quick and Morten Nome, and it's been a pleasure working with you.

I spent the academic year 2017-2018 at UC Berkeley, USA. In addition to all the interesting classes, projects and seminars, I want to acknowledge Roger Antonsen and all the new friends I made for making my time in California an amazing experience.

Last, but not least, I want to thank my family, my friends and my partner for always supporting my work, my interests and my path in life. I also want to thank them for all the non-cryptographic stuff we do together, making sure that my day to day life is (somewhat) balanced. Special thanks to Dahlia for being patient with me, and for proofreading my thesis.

Thank you!

PREFACE

This thesis builds on joint work with Herman Galteland. We co-authored a paper titled *Verifiable Random Secrets and Subliminal-Free Digital Signatures*, which is included in his Ph.D.-thesis [Gal20, VI]. Our paper defines a new concept called *Verifiable Random Secrets* (VRS), and gives two VRS-constructions: one based on the hardness of discrete logarithms, and another based on hard problems on lattices. We combine our VRS-constructions with Schnorr-like signatures to achieve *subliminal-free signatures* (SFS).

This thesis contains some, but not all, of the sections in [Gal20, VI]. Herman's main contribution was the schemes based on discrete logarithms, and my main contribution was the schemes based on lattices. The focus in this thesis is a slightly improved version of the lattice-based VRS- and SFS-schemes given in [Gal20, VI], in addition to a new, more generic and flexible VRS-construction based only on one-way functions. The new VRS offers smaller proofs but requires linear work in the security parameter.

Sections §1, §4 and §5 are co-authored with Herman Galteland [Gal20, VI]. Sections §2 and §3 are background material, where the shuffle-protocol in §3.2 is joint work with Diego Aranha, Carsten Baum, Kristian Gjøsteen and Thor Tunge [ABG$^+$, BGS]. Section §6 is my own contribution. Sections §4, §5 and §6 are the main new contributions in this thesis. We conclude in §7.

This thesis is available at `ntnuopen.ntnu.no` and `tjerandsilde.no`.

Tjerand Aga Silde
Trondheim, July 1st 2020

Contents

1 Introduction

A *subliminal channel* is a solution to Simmons's Prisoners' Problem [Sim84], where two prisoners want to communicate covertly over an overt channel controlled by a warden. The prisoners are allowed to send signed messages, to authenticate the sender of the message, but the messages themselves have to be sent in the clear. To communicate covertly the prisoners can create a subliminal channel, where the subliminal messages are encoded into the signatures. Only the prisoners, which may have some pre-shared secret knowledge, can recover the subliminal messages, while all signatures appear normal to everyone else.

The goal of the prisoners as subliminal sender and receiver is to communicate covertly, and the goal of the warden is to prevent any subliminal channel. In this thesis we will focus on achieving warden's goal of creating a subliminal-free signature scheme.

Constructing subliminal-free digital signature schemes for classical adversaries is solved [BS05, BGVS07, DX10, ZLLZ13]. Constructing subliminal-free digital signature schemes that are secure against a quantum adversary has been an open problem until now— designing such schemes is the main contribution from this thesis.

1.1 Warden Model

The subliminal sender `S` and receiver `R` want to reliably communicate discreetly over a communication channel controlled by a warden `W`. Before a signature is generated, `S` and `W` may interact to jointly produce a random value known to `S` but secret to `W`, which will be used to create a signature. `S` creates a proof that the random value was used in the signature. Then, `S` sends the message-signature-proof tuple to `W`, which verifies the signature and checks the proof. If, and only if, both are valid, then `W` forwards the message-signature pair to `R`. The proofs are only sent to `W` and cannot be used to send subliminal information.

The sender will be able to choose his own public and secret signing keys, however, S will not be able to update any keys during the signing process. Warden will abort if any signature is invalid with respect to the verification key of S, and then close the channel. Also, if S aborts during the signing process, e.g. if the signature does not include the subliminal bits and he wants to re-try, Warden closes the channel.

We assume that S and R may have shared secret information before they start communicating: e.g. a secret key for a suitable symmetric cryptosystem, and the signing key. The secret key can be used to encrypt the subliminal messages to make them indistinguishable from a random value, and the signing key can be used to recover subliminal messages. The sender is allowed to cheat during key generation to produce any desired signing key.

We are not interested in hiding information in the messages themselves, or steganography. We will assume that S is given a message to sign, and that the goal of S is to encode the subliminal messages into the signatures.

1.2 Subliminal-Free Digital Signatures

Our work builds upon the subliminal-free digital signature scheme with proof definition of Bohli et al. [BGVS07], where they constructed a subliminal-free variant of ECDSA. We give a similar definition of a subliminal-free digital signature scheme and construct two post-quantum subliminal-free digital signature schemes: one based on lattices, and a more generic construction based only on one-way functions.

We get a subliminal-free digital signature scheme if S is unable to choose any of the random values used to generate a signature. If S can choose the randomness, then he can easily replace the random values with the encryption of a subliminal message. Hence, we need a procedure for creating random values that is not controlled by S. We define a verifiable random secrets (VRS) scheme, and use the VRS together with Schnorr-like digital signature schemes to create a subliminal-free digital signature scheme. The

VRS is used by `S` and `W` to jointly generate a verifiable random number, which will be used to produce a signature. The sender also needs to include a proof showing that the random number generated in the VRS scheme was used to produce the signature.

A VRS is an interactive protocol between a prover and a verifier that produces verifiable random numbers known only to the prover and unpredictable for everyone else, along with proofs to convince the verifier that the randomness was generated honestly. VRSs are inspired by verifiable random functions (VRFs) by Micali et al. [MVR99]. The main difference between a VRF and a VRS is that the random number produced is secret to the verifier in a VRS but public in a VRF.

Schnorr-like digital signature schemes follow the same structure as zero-knowledge proofs of knowledge of opening of a commitment. The prover sends a new commitment of a random value to the verifier, the verifier replies with a challenge, and the sender generates a response using the challenge and the secret opening. This protocol can be made non-interactive using the Fiat-Shamir heuristic [FS87], where the challenge is generated by a hash function. If the message is a part of the input to the hash function, we get a digital signature scheme.

The lattice-based signature scheme of Lyubashevsky [Lyu09, Lyu12] follows the same pattern as Schnorr's digital signature scheme: commit, challenge, and response. The signer samples a random lattice-vector, computes the challenge using a hash function, and the response is generated using the random value, challenge, and secret signing key. Lyubashevsky's scheme uses rejection sampling to discard certain signatures, where a signature is sometimes rejected to make the signature distribution independent of the secret key.

1.3 Related Work

Simmons introduced the notion of subliminal channels as a solution to his prisoners' problem [Sim84]. Two partners in crime are arrested and put into separate parts of a jail. The prisoners wish to communicate with each other, to plan their escape, and the warden allows them to send messages if he can read the content of the messages sent, hoping to learn about any potential escape plan. The prisoners are allowed to sign their messages and can verify that they are sent from a prisoner and not from the warden. This is an authentication without secrecy communication channel controlled by the warden. The problem of the prisoners is to make a subliminal channel that stays undetected by the warden, and the problem of the warden is to prevent any subliminal channels. Simmons showed that subliminal channels in digital signature schemes exist [Sim85, Sim86, Sim94] and since more have been found [AVPN96, BS05, ZL08, LWZG10, HAZ17, GG19].

Desmedt was the first to construct and locate subliminal-free digital signature schemes [Des88], which since has been continued [Sim93, Sim94, Sim98, BS05, BGVS07, DX10, ZLLZ13]. Bohli et al. introduced the notion of a *subliminal-free with proof signature scheme*, where the sender of a signature sends a proof to the warden, and only to the warden, that proves the signature sent is subliminal-free [BGVS07]. In *divertible protocols* [OO90, BD91, BBS98, BDI+99] a third party can be inserted between the two communicators, where the third party can remove or detect subliminal messages. A *cryptographic reverse firewall* [MSD14, CMY+16] sits around a user's computer and modifies messages to maintain the usability of the computer; preserve security of the protocol generating the message, and prevent information from leaking to the outside world.

NIST's post-quantum standardization project has asked for digital signature schemes [NIST17], and not all digital signature schemes accepted to the second round are subliminal free [GG19]. The CRYSTAL-Dilithium [DKL+] and qTesla [ABB+20] submissions are similar to Schnorr signatures and can become subliminal-free using our techniques.

2 Preliminaries

2.1 Notation

Let S be a set and $\texttt{Alg}(\cdot)$ an algorithm. Then, by $s \leftarrow \texttt{Alg}(\cdot)$ we mean that s is assigned the output of $\texttt{Alg}(\cdot)$, by $s \xleftarrow{\$} S$ we mean that s is assigned an uniformly random element of S (unless a specific distribution is specified), and by $s \leftarrow s'$ we mean that s is assigned the value s'. Let λ be the security parameter, then $\epsilon(\lambda)$ is a negligible function in the security parameter.

2.2 Polynomial Rings and Norms

Let $p, r \in \mathbb{N}^+$ and $N = 2^r$. Then we define the rings $R = \mathbb{Z}[X]/\langle X^N + 1\rangle$ and $R_p = R/\langle p\rangle$, that is, R_p is the ring of polynomials modulo $X^N + 1$ with integer coefficients modulo p. We define the norms of elements

$$f(X) = \sum \alpha_i X^i \in R$$

to be the norms of the coefficient vector as a vector in $\mathbb{Z}^N$:

$$||f||_1 = \sum |\alpha_i| \qquad ||f||_2 = \left(\sum \alpha_i^2\right)^{1/2} \qquad ||f||_\infty = \max_{i\in\{1,\ldots,n\}}\{|\alpha_i|\}.$$

For an element $\bar{f} \in R_p$ we choose coefficients as the representatives in $\left[-\frac{p-1}{2}, \frac{p-1}{2}\right]$, and then compute the norms as if $\bar{f}$ is an element in R. For vectors $\boldsymbol{a} = (a_1, \ldots, a_k) \in R^k$ we define the 2-norm to be

$$\|\boldsymbol{a}\|_2 = \sqrt{\sum \|a_i\|_2^2},$$

and analogously for the ∞-norm. We omit the subscript for the 2-norm.

2.3 Short elements in R_p

It can be seen from Corollary 1.2 in [LS18] that sufficiently short elements in R_p are invertible. In the following, we assume for simplicity that the parameters are set such that all non-zero elements of ∞-norm at most 2 are invertible in R_p. We furthermore define

$$\mathcal{C} = \{c \in R_p \mid \|c\|_\infty = 1, \|c\|_1 = \nu\},$$

which consists of all elements in R_p that have trinary coefficients and are non-zero in exactly ν positions. This means that for any two distinct $c, c' \in \mathcal{C}$, the difference $c - c'$ is invertible as well. For convenience, denote by

$$\bar{\mathcal{C}} = \{c - c' \mid c \neq c' \in \mathcal{C}\}$$

the set of such differences.

2.4 Discrete Gaussian Distribution

The continuous normal distribution over $\mathbb{R}^k$ centered at $\boldsymbol{v} \in \mathbb{R}^k$ with standard deviation σ is given by

$$\rho(\boldsymbol{x})^N_{\boldsymbol{v},\sigma} = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-||\boldsymbol{x} - \boldsymbol{v}||^2}{2\sigma^2}\right).$$

When sampling randomness for our lattice-based commitment scheme, we'll need samples from the *discrete Gaussian distribution*. This distribution is achieved by normalizing the continuous distribution over R^k by letting

$$\mathcal{N}^k_{\boldsymbol{v},\sigma}(\boldsymbol{x}) = \frac{\rho^{kN}_{\boldsymbol{v},\sigma}(\boldsymbol{x})}{\rho^{kN}_\sigma(R^k)},$$

where $\boldsymbol{x} \in R^k$ and $\rho^{kN}_\sigma(R^k) = \sum_{\boldsymbol{x} \in R^k} \rho^{kN}_\sigma(\boldsymbol{x})$. When $\sigma = 1$ or $\boldsymbol{v} = \boldsymbol{0}$, they are omitted.

The most efficient way to sample elements from a discrete Gaussian distribution is using rejection sampling. Rejection sampling is a technique used to sample from a distribution Φ_0, using samples from a similar distribution Φ_1. Let Φ_0 and Φ_1 be two distributions defined over the same domain S, whose probability mass functions are efficiently computable. Let M be a constant such that $\Phi_0(x) \leq M \cdot \Phi_1(x)$ for all $x \in S$. Then the distribution generated by the algorithm in Figure 1 will output samples distributed according to Φ_0 while only using samples from Φ_1 and a uniform distribution. The performance of the algorithm depends strongly on how similar Φ_0 and Φ_1 are, in particular, how small the scaling factor M is.

Rejection Sampling

Input: distributions Φ_0, Φ_1 and constant M s.t. $\Phi_0(x) \leq M \cdot \Phi_1(x) \forall x$

Output: x from the distribution Φ_0

1 : $x \xleftarrow{\$} \Phi_1$

2 : $u \xleftarrow{\$} [0, 1]$

3 : **if** $u \leq \frac{\Phi_0(x)}{M \cdot \Phi_1(x)}$ **then** goto 1

4 : **Return:** x

Figure 1: Rejection sampling algorithm.

However, rejection sampling can be very expensive, and it's a randomized procedure. If we do some pre-computations, we can use the algorithm given in qTelsa [ABB+20] to generate the discrete Gaussians deterministically from a random seed. This is specified in Algorithm 11 in the NIST-submission of qTelsa [AAB+19]. We will use the rejection sampling when generating Gaussians for the commitments, but use the deterministic algorithm to generate Gaussians used in the lattice-based VRS.

2.5 The k-SUM Problem

The *k-sum problem* (k-SUM) is a variant of the subset sum problem (SSP). SSP is a NP-complete decision problem where given a set of n integers $a_1, a_2, \ldots, a_n$ and a number S; is there a non-empty subset of $a_1, a_2, \ldots, a_n$ whose sum is S? This decision problem is as hard as its search-equivalent.

The k-SUM problem is the problem of deciding if there is a subset of size k of $a_1, a_2, \ldots, a_n$ whose sum is S. It can easily be shown that SSP reduces to k-SUM, as a polynomial time k-SUM-solver easily could be used to solve SSP by trying $k = 1, 2, \ldots, n$ until it finds a solution. Further, the decision variant of k-SUM is as hard as the search variant of k-SUM, as one could find the subset of size k by removing elements individually and checking if there is a solution or not for the new set. Given n and k, the fastest algorithm for solving k-SUM runs in $\mathcal{O}(n^{k/2})$ [Eri95].

2.6 Subliminal-Free with Proof Signature Scheme

We include the following definition of Bohli et al. [BGVS07] for comparison, where we give our own definition of a subliminal-free signature scheme in Section 5.3 followed by a note on the differences between the definitions.

Definition 1 (Subliminal-Free with Proof Signature Scheme [BGVS07])**.** *A subliminal-free with proof signature scheme is a quintuple of algorithms $(\mathcal{K}, \mathcal{K}_{SF}, \mathcal{S}, \mathcal{V}, \mathcal{C})$, where*

- *The* key generation algorithm $\mathcal{K}$ *takes the security parameter λ as input and returns a pair of verification and signing keys (vk, sk).*
- *The* subliminal-free key generation algorithm $\mathcal{K}_{SF}$ *takes vk and sk as input and generates the information ci that the warden needs to check the signature computation.*
- *The* signing algorithm $\mathcal{S}$ *takes a message m and the signing key sk as input and produces a signature σ of m under vk and a proof t.*

- *The* verification algorithm $\mathcal{V}$ *takes a message* m*, a signature* σ *and the public verification key* vk *as input and returns* 1 *if* σ *is a valid signature for* m *with respect to* vk*, and* 0 *otherwise.*
- *The* checking algorithm $\mathcal{C}$ *takes a message* m*, a signature* σ*, a verification key* vk*, the checking information* ci *and a proof* t *as input, and returns* 1 *if* $\mathcal{V}(m, \sigma, vk) = 1$ *and* (σ, t) *is a valid output of* $\mathcal{S}(m, sk)$*.*

Moreover, for any algorithm A *taking the security parameter* λ *as input, the probability of giving as output values* $vk, sk, ci, m, \sigma_1, \sigma_2, t_1, t_2$ *such that* $(vk, sk), ci$ *are computationally indistinguishable from the output of* $\mathcal{K}$ *and* $\mathcal{K}_{SF}$*, respectively,* $\sigma_1 \neq \sigma_2$ *and* $\mathcal{C}(m, \sigma_1, vk, ci, t_1) = \mathcal{C}(m, \sigma_2, vk, ci, t_2) = 1$ *is negligible in the security parameter* λ*.*

2.7 Commitment Schemes

Commitment schemes were first introduced by Blum [Blu83], and have since become an essential component in many advanced cryptography protocols.

Definition 2 (Commitment Scheme)**.** *A commitment scheme consists of three algorithms: key generation* (`KeyGen`)*, commitment* (`Com`) *and opening* (`Open`)*, where*

- `KeyGen`*, on input security parameter* 1^λ*, outputs public parameters* `pp`*,*
- `Com`*, on input message* m*, outputs commitment* c *and randomness* r*,*
- `Open`*, on input* m*,* c *and* r*, outputs either* 0 *or* 1*,*

and the public parameters `pp` *are implicit input to* `Com` *and* `Open`*.*

Definition 3 (Completeness)**.** *We say that the commitment scheme is* complete *if an honestly generated commitment is accepted by the opening algorithm. Hence, we want that*

$$\Pr\left[\texttt{Open}(m, c, r) = 1 \;:\; \begin{array}{c}\texttt{pp} \leftarrow \texttt{KeyGen}(1^\lambda) \\ (c, r) \leftarrow \texttt{Com}(m)\end{array}\right] = 1.$$

Definition 4 (Hiding)**.** *We say that a commitment scheme is* hiding *if an adversary* `A`*, after giving two messages* m_1 *and* m_2 *to a commitment*

oracle $\mathtt{O}_{\mathtt{com}}$ *and receiving the commitment* c *to either* m_1 *or* m_2 *(chosen at random), cannot distinguish which message* c *is a commitment to. Hence, we want that*

$$2 \cdot |\Pr\left[b = b' \ : \ \begin{array}{c} \mathtt{pp} \leftarrow \mathtt{KeyGen}(1^\lambda) \\ (m_1, m_2) \leftarrow \mathtt{A}(\mathtt{pp}) \\ b \xleftarrow{\$} \{0,1\}, c \leftarrow \mathtt{O}_{\mathtt{com}}(m_b) \\ b' \leftarrow \mathtt{A}(c) \end{array}\right] - \frac{1}{2}| \leq \epsilon(\lambda).$$

Definition 5 (Binding)**.** *We say that a commitment scheme is* binding *if an adversary* $\mathtt{A}$*, after creating a commitment* c *to a messages* m*, cannot find a valid opening of* c *to a different message* $\hat{m}$*. Hence, we want that*

$$\Pr\left[\begin{array}{c} m \neq \hat{m} \\ \mathtt{Open}(m, c, r) = 1 \\ \mathtt{Open}(\hat{m}, c, \hat{r}) = 1 \end{array} \ : \ \begin{array}{c} \mathtt{pp} \leftarrow \mathtt{KeyGen}(1^\lambda) \\ m \leftarrow \mathtt{A}(\mathtt{pp}) \\ (c, r) \leftarrow \mathtt{A}(m) \\ (\hat{m}, \hat{r}) \leftarrow \mathtt{A}(m, c, r) \end{array}\right] \leq \epsilon(\lambda).$$

Definition 6 (Unconditional and Computational Adversaries)**.** *We say that a commitment scheme is* unconditionally hiding *(*unconditionally binding*) if the scheme is hiding (binding) against an unbounded adversary, and we say that it is* computationally hiding *(*computationally binding*) if the scheme is hiding (binding) against a bounded probabilistic time adversary.*

2.8 Digital Signature Schemes

Definition 7 (Digital Signature Schemes)**.** *A digital signature scheme consists of three algorithms: key generation* (`KeyGen`)*, signing* (`Sign`) *and verification* (`Verify`)*, where*

- `KeyGen`*, on input the security parameter* 1^λ*, outputs public parameters* `pp`*, a signing key* `sk`*, and a verification key* `vk`*,*
- `Sign`*, on input a message* m *and* `sk`*, outputs a signature* σ*,*
- `Verify`*, on input* m*,* σ *and* `vk`*, outputs either* 0 *or* 1*,*

and the public parameters `pp` *are implicit inputs to* `Sign` *and* `Verify`*.*

We require the digital signature scheme to be *complete* (sometimes also referred to as *correct*), and to be secure against existential forgery under an adaptive chosen message attack, following the definitions from Goldwasser et al. [GMR88].

Definition 8 (Completeness)**.** *We say that the digital signature scheme is* complete *if honestly generated signatures are accepted by the verification algorithm. Hence, we want that*

$$\Pr\left[\mathtt{Verify}(m,\sigma,\mathtt{vk})=1 \;:\; \begin{matrix}(\mathtt{pp},\mathtt{sk},\mathtt{vk})\leftarrow\mathtt{KeyGen}(1^\lambda)\\(\sigma)\leftarrow\mathtt{Sign}(m,\mathtt{sk})\end{matrix}\right]=1.$$

Definition 9 (Existential Forgeability)**.** *We say that the digital signature scheme is secure against* existential forgeability *if an adversary* $\mathtt{A}$*, after given valid signatures* σ_i *of messages* m_i *of* $\mathtt{A}$*'s choice from a signing oracle* $\mathtt{O}_{\mathtt{sign}}$*, cannot forge a signature on any new message under the same public-private key pair. Hence, we want that*

$$\Pr\left[\begin{matrix}\hat{m}\notin\{m_i\}\\ \mathtt{Verify}(\hat{m},\hat{\sigma},\mathtt{vk})=1\end{matrix} \;:\; \begin{matrix}(\mathtt{pp},\mathtt{vk})\leftarrow\mathtt{KeyGen}(1^\lambda)\\(\hat{m},\hat{\sigma})\leftarrow\mathtt{A}^{\mathtt{O}_{\mathtt{sign}}}(\mathtt{pp},\mathtt{vk},\{m_i\})\end{matrix}\right]\leq\epsilon(\lambda).$$

Here $\{m_i\}$ *is the set of messages signed by the signing oracle* $\mathtt{O}_{\mathtt{sign}}$*.*

2.9 Zero-Knowledge Proofs

These definitions are based on Goldwasser et al. [GMR85]. Let $\mathtt{L}$ be a language, and let $\mathtt{R}$ be a relation on $\mathtt{L}$. Then, x is an element in $\mathtt{L}$, if there exists a witness w such that $(x,w)\in\mathtt{R}$.

Definition 10 (Zero-Knowledge Proofs)**.** *An interactive zero-knowledge proof protocol* Π *consists of two parties: a prover* $\mathtt{P}$ *and a verifier* $\mathtt{V}$*, and a setup algorithm* ($\mathtt{Setup}$)*, where* $\mathtt{Setup}$*, on input the security parameter* 1^λ*, outputs public setup parameters* $\mathtt{sp}$*. The protocol consists of a transcript* $\mathtt{T}$ *of the communication between* $\mathtt{P}$ *and* $\mathtt{V}$*, with respect to* $\mathtt{sp}$*, and the conversation terminates with* $\mathtt{V}$ *outputting either* 1 *or* 0*. Let* $\langle\mathtt{P}(\mathtt{sp},x,w),\mathtt{V}(\mathtt{sp},x)\rangle$ *denote the output of* $\mathtt{V}$ *on input* x *after its interaction with* $\mathtt{P}$*, who has witness* w*.*

Definition 11 (Completeness)**.** *We say that a zero-knowledge proof protocol* Π *is* complete *if* $\mathtt{V}$ *outputs* 1 *when* $\mathtt{P}$ *knows a witness* w*. Hence, for any sampling algorithm* $\mathtt{P}_0$ *we want that*

$$\Pr\left[\langle \mathtt{P}(\mathtt{sp}, x, w), \mathtt{V}(\mathtt{sp}, x)\rangle = 1 \ : \ \begin{array}{c} \mathtt{sp} \leftarrow \mathtt{Setup}(1^\lambda) \\ (x, w) \leftarrow \mathtt{P}_0(\mathtt{sp}) \\ (x, w) \in \mathtt{R} \end{array}\right] = 1.$$

Definition 12 (Soundness)**.** *We say that a zero-knowledge proof protocol* Π *is* sound *if a cheating prover* $\mathtt{P}^*$ *that does not know a witness cannot convince* $\mathtt{V}$*. Hence, for any* x *not in the language* $\mathtt{L}$

$$\Pr\left[\langle \mathtt{P}^*(\mathtt{sp}, x, \cdot), \mathtt{V}(\mathtt{sp}, x)\rangle = 1 \ : \ \begin{array}{c} \mathtt{sp} \leftarrow \mathtt{Setup}(1^\lambda) \\ \forall x \notin \mathtt{L} \end{array}\right] \leq \frac{1}{2}.$$

Definition 13 (Honest-Verifier Zero-Knowledge)**.** *We say that a zero-knowledge proof protocol* Π *is* honest-verifier zero-knowledge *if a honest but curious verifier* $\mathtt{V}^*$ *that follows the protocol cannot learn anything beyond the fact that* $x \in \mathtt{L}$*. Hence, we want for real accepting transcripts* $\mathtt{T}_{\langle \mathtt{P}(\mathtt{sp},x,w),\mathtt{V}(\mathtt{sp},x)\rangle}$ *between a prover* $\mathtt{P}$ *and a verifier* $\mathtt{V}$*, and a accepting transcript* $\mathtt{S}_{\langle \mathtt{P}(\mathtt{sp},x,\cdot),\mathtt{V}(\mathtt{sp},x)\rangle}$ *generated by simulator* $\mathtt{S}$ *that only knows* x*, that*

$$2 \cdot |\Pr\left[b = b' \ : \ \begin{array}{c} \mathtt{sp} \leftarrow \mathtt{Setup}(1^\lambda) \\ \mathtt{T}_1 = \mathtt{T}_{\langle \mathtt{P}(\mathtt{sp},x,w),\mathtt{V}(\mathtt{sp},x)\rangle} \leftarrow \Pi(\mathtt{sp}, x, w) \\ \mathtt{T}_2 = \mathtt{S}_{\langle \mathtt{P}(\mathtt{sp},x,\cdot),\mathtt{V}(\mathtt{sp},x)\rangle} \leftarrow \mathtt{S}(\mathtt{sp}, x) \\ b \xleftarrow{\$} \{0,1\}, \mathtt{T}' \leftarrow \mathtt{T}_b \\ b' \leftarrow \mathtt{V}^*(\mathtt{T}', \mathtt{sp}, x) \end{array}\right] - \frac{1}{2}| \leq \epsilon(\lambda).$$

2.10 Verifiable Random Functions

We give the definition of a verifiable random function based on the work by Micali et al. [MVR99].

Definition 14 (Verifiable Random Functions)**.** *A verifiable random function scheme consists of three algorithms: key generation* (`KeyGen`)*, function evaluation* (`Eval`) *and verification* (`Verify`)*, where*

- `KeyGen`*, on input the security parameter* 1^λ*, outputs a public function* `f`*, an evaluation key* `sk`*, and a verification key* `vk`*,*

- $\texttt{Eval}$, *on input an element* x *and* $\texttt{sk}$, *outputs an evaluation* $y = \texttt{f}(\texttt{sk}, x)$ *and a proof* π,
- $\texttt{Verify}$, *on input* x, y, π *and* $\texttt{vk}$, *outputs either* 0 *or* 1,

and the public function $\texttt{f}$ *is an implicit input to* $\texttt{Eval}$ *and* $\texttt{Verify}$.

Definition 15 (Completeness)**.** *We say that a verifiable random function scheme is* complete *if the verification algorithm always accepts the result of a honest evaluation of the function. Hence, we want that*

$$\Pr\left[\texttt{Verify}(x, y, \pi, \texttt{vk}) = 1 \;:\; \begin{matrix}(\texttt{f}, \texttt{sk}, \texttt{vk}) \leftarrow \texttt{KeyGen}(1^\lambda) \\ (y, \pi) \leftarrow \texttt{Eval}(x, \texttt{sk})\end{matrix}\right] = 1.$$

Definition 16 (Uniqueness)**.** *We say that a verifiable random function scheme is* uniquely provable *if an adversary* $\texttt{A}$*, after creating an evaluation* y *of* x *together with a proof* π*, cannot find another valid evaluation* $\hat{y}$ *and proof* $\hat{\pi}$ *to* x*. Hence, we want that*

$$\Pr\left[\begin{matrix}y \neq \hat{y} \\ \texttt{Verify}(x, y, \pi, \texttt{vk}) = 1 \\ \texttt{Verify}(x, \hat{y}, c, \hat{\pi}, \texttt{vk}) = 1\end{matrix} \;:\; \begin{matrix}(\texttt{f}, \texttt{sk}, \texttt{vk}) \leftarrow \texttt{KeyGen}(1^\lambda) \\ (x, y, \pi, \hat{y}, \hat{\pi}) \leftarrow \texttt{A}(\texttt{f}, \texttt{sk}, \texttt{vk})\end{matrix}\right] \leq \epsilon(\lambda).$$

Definition 17 (Pseudorandomness)**.** *We say that a verifiable random function scheme is* pseudorandom *if an adversary* $\texttt{A}$*, after given valid evaluations* y_i *with proofs* π_i *of inputs* x_i *of* $\texttt{A}$*'s choice from an evaluation oracle* $\texttt{O}_{\texttt{eval}}$*, for a known* $\texttt{f}$*, cannot distinguish if a value* y *is a valid evaluation of a* x *of* $\texttt{A}$*'s choice with respect to* $\texttt{f}$*, or if* y *is a random string. Hence, we want that*

$$2 \cdot |\Pr\left[\begin{matrix}x \notin \{x_i\} \\ b = b'\end{matrix} \;:\; \begin{matrix}(\texttt{f}, \texttt{sk}, \texttt{vk}) \leftarrow \texttt{KeyGen}(1^\lambda) \\ \{(y_i, \pi_i)\} \leftarrow \texttt{A}^{\texttt{O}_{\texttt{eval}}(\texttt{f}, \texttt{vk}, \{x_i\})} \\ x \leftarrow \texttt{A}(\texttt{f}, \texttt{vk}, \{(x_i, y_i, \pi_i)\}), \\ (y', \pi) \leftarrow \texttt{Eval}(x, \texttt{sk}), \hat{y} \xleftarrow{\$} \{0,1\}^{\texttt{len}(y')} \\ b \xleftarrow{\$} \{0,1\}, y \leftarrow by' + (1-b)\hat{y} \\ b' \leftarrow \texttt{A}(\texttt{f}, x, y)\end{matrix}\right] - \frac{1}{2}| \leq \epsilon(\lambda).$$

3 Lattice-Based Cryptography

First, we present the lattice-problems Ring Learning With Errors (R-LWE) and Ring Short Integer Solutions (R-SIS). Then we introduce the building blocks of our lattice-based verifiable random secret scheme and subliminal-free signature scheme, upon which hardness relies on R-LWE and R-SIS.

3.1 Hard Problems on Lattices

The security of the following schemes is based on the hardness of the Ring Learning With Errors (R-LWE) problem and the Ring Short Integer Solutions (R-SIS) problem.

Definition 18 (R-LWE)**.** *The* Ring Learning with Errors *problem over* R_p *is the following:*

1. *Draw uniform* $a \xleftarrow{\$} R_p$
2. *Draw discrete Gaussians* $s, e \leftarrow \mathcal{N}_\sigma$
3. *Compute* $b = as + e$ *and publish* (a, b)

The decisional *problem is to distinguish* b *from uniformly random, and the* computational *problem is to find* s *and* e *where both are* short*.*

Definition 19 (R-SIS)**.** *The* Ring Short Integer Solutions *problem over* R_p *is the following:*

1. *Draw uniform* $a \xleftarrow{\$} R_p$
2. *Draw short* $e \xleftarrow{\$} R_p$ *s.t.* $||e||_\infty \leq B$
3. *Compute* $b = ae$ *and publish* (a, b)

The decisional *problem is to distinguish* b *from uniformly random, and the* computational *problem is to find* e *where* e *is bounded by* B*.*

3.2 Commitments

We briefly present the lattice-based commitment scheme by Baum et al. [BDL+18], which offers an efficient zero-knowledge proof of linear relations that will be useful when combining the VRS and the signature schemes to achieve subliminal-free signatures.

Definition 20 (Lattice-Based Commitments [BDL+18]). *Let R_p be the ring of polynomials modulo $X^N + 1$ with integer coefficients modulo p. The lattice-based commitment scheme consists of three algorithms: key generation* (`KeyGen`)*, commitment* (`Com`) *and opening* (`Open`)*, where*

- `KeyGen`*, on input the security parameter 1^λ, outputs a public matrix $\boldsymbol{A}$ such that*

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{a}' \end{bmatrix} = \begin{bmatrix} 1 & a_1 & a_2 \\ 0 & 1 & a_3 \end{bmatrix}, \text{where } a_1, a_2, a_3 \xleftarrow{\$} R_p,$$

- `Com`*, on input a message $m \in R_p$, samples an $\boldsymbol{r} \xleftarrow{\$} R_p^3$ where $\|\boldsymbol{r}\|_\infty = 1$, and computes*

$$\boldsymbol{c} = \texttt{Com}(m; \boldsymbol{r}) = \boldsymbol{A} \cdot \boldsymbol{r} + \begin{bmatrix} 0 \\ m \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix},$$

  and returns $\boldsymbol{c}$ and $\boldsymbol{d} = (m; \boldsymbol{r}, 1)$,

- `Open`*, on input $(m, \boldsymbol{r}, f)$ with $f \in \bar{\mathcal{C}}$, verifies the opening by checking if*

$$f \cdot \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \stackrel{?}{=} \boldsymbol{A} \cdot \boldsymbol{r} + f \cdot \begin{bmatrix} 0 \\ m \end{bmatrix},$$

  and that $\|r_i\| \leq 4\sigma\sqrt{N}$ for $\boldsymbol{r} = (r_0, r_1, r_2)$ with $\sigma = 11 \cdot \nu \cdot \sqrt{3N}$. It outputs 1 if all these conditions hold, and 0 otherwise.

3.3 Zero-Knowledge Proofs

We present two zero-knowledge protocols that our lattice-based verifiable random secret scheme depends on to preserve privacy and soundness.

Zero-Knowledge Proof of Linear Relations Define the following three commitments:

$$[x_1] = \texttt{Com}(x_1; \boldsymbol{r}) = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix},$$

$$[x_2] = \mathtt{Com}(x_2; \boldsymbol{r}') = \begin{bmatrix} c_1' \\ c_2' \end{bmatrix},$$
$$[x_3] = \mathtt{Com}(x_3; \boldsymbol{r}'') = \begin{bmatrix} c_1'' \\ c_2'' \end{bmatrix}.$$

Let $[x_1], [x_2]$ and $[x_3]$ be such that $x_3 = \alpha_1 x_1 + \alpha_2 x_2$ for some public values $\alpha_1, \alpha_2 \in R_p$. Then the Π_{Lin}-protocol in Figure 2 is a zero-knowledge proof of knowledge (ZKPoK) of this relation (an adapted version of the linearity proof in [BDL+18]), and the Π_{LinV}-protocol in Figure 3 is the verification algorithm for the proof. This protocol can easily be extended to prove linear relations between an arbitrary number of commitments.

Further, let $\pi_L \leftarrow \Pi_{\text{Lin}}(([x_1], [x_2]), [x_3], (\alpha_1, \alpha_2))$ denote the run of the Π_{Lin}-protocol to prove the relation $x_3 = \alpha_1 x_1 + \alpha_2 x_2$ producing a proof $\pi_L = ((t, t', t''), \beta, (\boldsymbol{z}, \boldsymbol{z}', \boldsymbol{z}''))$. If $\alpha_1 = \alpha_2 = 1$, then the scalars are omitted from the input. Let $0 \vee 1 \leftarrow \Pi_{\text{LinV}}(([x_1], [x_2]), [x_3], (\alpha_1, \alpha_2), \pi_L)$ denote the verification of this proof.

Zero-Knowledge Proof of Correct Shuffle In the work by Silde et al. [ABG+, BGS] they give an efficient protocol Π_{Shuffle} for a Neff-like [Nef01] shuffle of known values for the lattice-based commitments by Baum et al. [BDL+18]. Given a list of elements $(\hat{M}_1, \hat{M}_2, \ldots, \hat{M}_\tau)$ from R_p and commitments $([M]_1, [M]_2, \ldots, [M]_\tau)$, we can prove that the $[M]_i$'s are commitments to the $\hat{M}_{\gamma(i)}$'s, for some secret permutation γ of the indices.

Let $\pi_S \leftarrow \Pi_{\text{Shuffle}}(\{[M]_i\}, \{\hat{M}_i\}, \gamma)$ denote the run of the shuffle-protocol, with proof π_S. Let $0 \vee 1 \leftarrow \Pi_{\text{ShuffleV}}(\{[M]_i\}, \{\hat{M}_i\}, \pi_S)$ denote the verification of this proof. Note that we can think of $\{\hat{M}_i\}$ as commitments with randomness zero. Note that since $\{\hat{M}_i\}$ are known values, we can give a simpler proof for each linear relation. Each equation is of the form $x_3 = \alpha_1 x_1 + \alpha_2$, which reduces the proof size by one element for both the commit and the response phase of the scheme. See [ABG+] for more details.

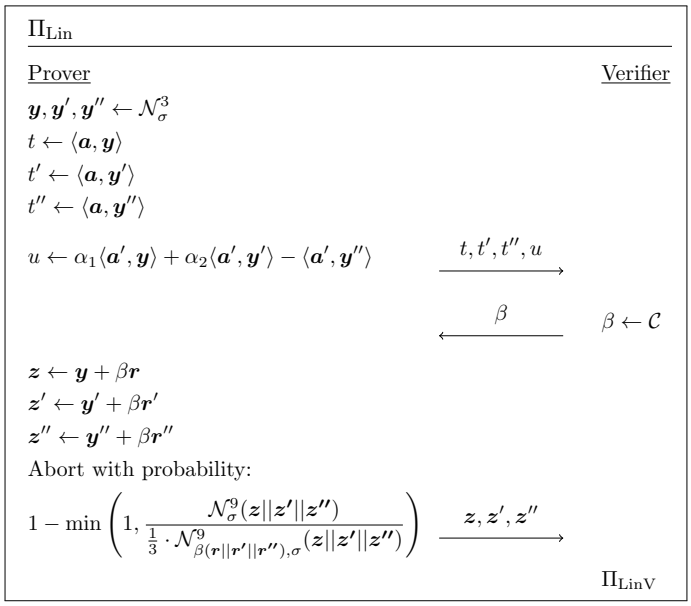

Figure 2: Zero-knowledge proof of knowledge protocol Π_{Lin} for the relation $x_3 = \alpha_1 x_1 + \alpha_2 x_2$, given commitments $[x_1], [x_2], [x_3]$ and scalars α_1, α_2.

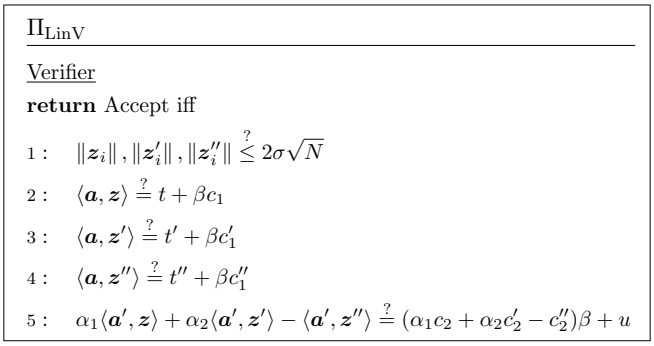

Figure 3: Verification protocol Π_{LinV} for the Π_{Lin}-protocol.

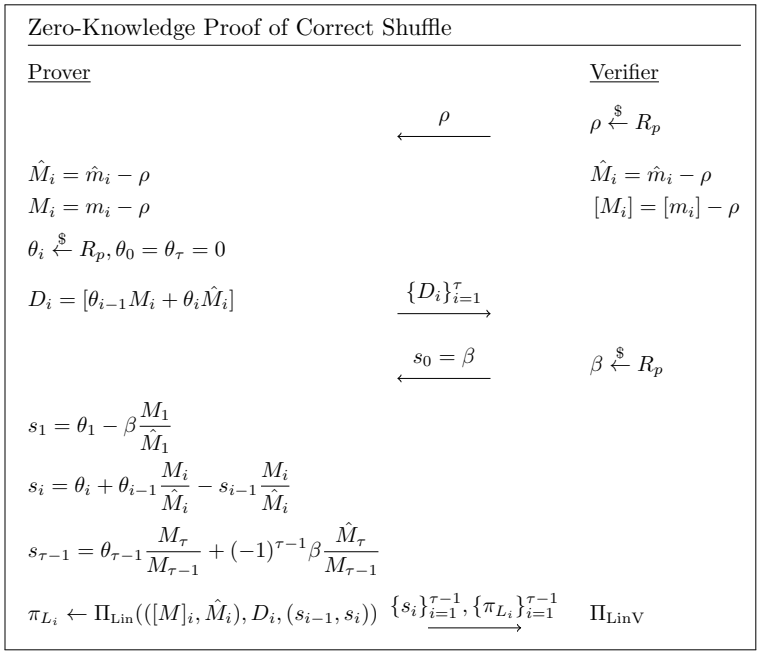

Figure 4: The zero-knowledge protocol of correct shuffle.

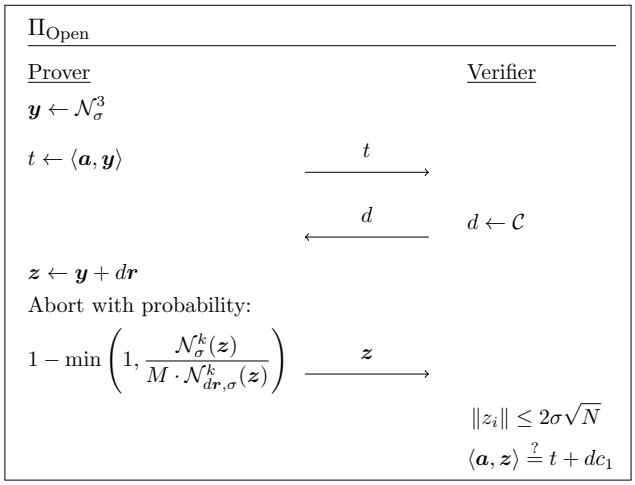

Figure 5: Zero-Knowledge Proof of Knowledge of Opening of $\boldsymbol{c} = \texttt{Com}(x; \boldsymbol{r})$.

3.4 Signatures

The most efficient lattice based signature schemes are based on the Schnorr-like [Sch89] signatures by Lyubashevsky [Lyu09, Lyu12]. Furthermore, the zero-knowledge proof of opening given by Baum et al. [BDL+18] follows this exact structure. Let R_p be the message space, let H be a hash function $H : R_p \times R_p \to \mathcal{C}$ and let $\texttt{vk} = \boldsymbol{c}$ be the public verification key, where $\boldsymbol{c} = \texttt{Com}(0; \boldsymbol{r})$ is a commitment to 0 with randomness $\boldsymbol{r}$, and let the secret key be $\texttt{sk} = \boldsymbol{r}$. Then the protocol Π_{Open} in Figure 5 can be turned into a signature scheme by applying the Fiat-Shamir transform where $d = H(t, m)$, for a message $m \in R_p$. Let $\sigma = (t, \boldsymbol{z}) \leftarrow \texttt{Sign}(m, \texttt{sk})$ denote the running of the signature algorithm with signature σ; furthermore, let $0 \vee 1 \leftarrow \texttt{Verify}(\texttt{pk}, m, \sigma)$ denote the verification of this signature.

4 Verifiable Random Secrets

To create a subliminal-free digital signature scheme, we need a procedure to output verifiable random numbers. A verifiable random function (VRF) has a lot of useful properties; however, the constructions require making the output public for anyone to verify. This means it is inapplicable to digital signature schemes, which require the randomness to be secret to create secure signatures. On the other hand, the verifier must be able to verify that the random value is generated in some certain unpredictable way, to prevent the prover intentionally choosing randomness to their advantage. One could imagine this being done in a zero-knowledge protocol. The challenge in this case is that we want to generate some random element that can be used later, and hence, we cannot just prove that we have generated a random element – we also have to provide information that can be included in the subsequent application.

A verifiable random secret (VRS) scheme is an interactive protocol where the prover wants to produce a secret random value, and publish a commitment of that value together with a proof to convince the verifier that the committed value is randomly generated. This can be done in the following way: first the prover commits to a random value and sends the commitment to the verifier. The verifier then returns a random challenge to the prover, which he in turn uses to generate a final commitment and a proof. The commitment contains a random value which was unpredictable for the prover until he got the challenge, and is secret to the verifier even after the protocol is completed. Anyone with access to the final commitment and the proof can check that the commitment was generated in a proper way, and hence, will be convinced that the content is random.

We want the VRS to have similar security properties as the VRF and the ZKP. Therefore, we adapt some of the security notions from VRFs and ZKPs, but change them slightly to fit the new setting.

Definition 21 (Verifiable Random Secrets)**.** *A Verifiable Random Secret-scheme (VRS) consists of a function $r(\cdot,\cdot)$, an interactive protocol seed*

($\Pi_{\mathtt{Seed}}$) *and five algorithms: setup* (`Setup`)*, commit* (`Com`)*, challenge* (`Challenge`)*, generation* (`Generate`) *and check* (`Check`)*, where*

- `Setup`*, on input security parameter* 1^λ*, outputs public parameters* `sp`*,*
- $\Pi_{\mathtt{Seed}}$*, on input* `sp`*, outputs a random seed* s*,*
- `Com`*, on input seed* s*, outputs commitment* $\tilde{c}$ *of* s *and opening* $\tilde{d}$*,*
- `Challenge`*, on no input, outputs a random challenge* t*,*
- `Generate`*, on input commitment* $\tilde{c}$*, opening* $\tilde{d}$ *and challenge* t*, outputs commitment* c*, opening* d *of* c *(containing* $r = r(s,t)$*) and proof* π*,*
- `Check`*, on input* $\tilde{c}$ *and* c*, challenge* t*, and proof* π*, outputs* 0 *or* 1*,*

and the public parameters `sp` *are implicit inputs to all algorithms following* `Setup`*.*

Remark 1. *In the interactive protocol* $\Pi_{\mathtt{Seed}}$ *there may be one or more participants. Any participant of the protocol may be dishonest during the seed generation. The output seed may be given to some or all participants.*

The first thing we require from a VRS is that it is complete.

Definition 22 (Completeness)**.** *We say that the VRS is* complete *if a prover* `P` *always can convince a honest verifier* `V` *that a honestly generated proof is valid. Hence, we want that*

$$\Pr\left[\mathtt{Check}(\tilde{c},t,c,\pi)=1 \;:\; \begin{array}{c} \mathtt{sp}\leftarrow\mathtt{Setup}(1^\lambda) \\ s\leftarrow\Pi_{\mathtt{Seed}}(\mathtt{sp}) \\ (\tilde{c},\tilde{d})\leftarrow\mathtt{Com}(s) \\ t\leftarrow\mathtt{Challenge}(\cdot) \\ (c,d,\pi)\leftarrow\mathtt{Generate}(\tilde{c},\tilde{d},t) \end{array}\right]=1.$$

The interactive VRS is visualized in Figure 6. Further, we use games to define the security of the scheme, and continue by defining *Binding*, *Prover bit-Unpredictability*, *Verifier Secrecy* and *Honest-Verifier Secrecy* for a VRS.

Definition 23 (Binding)**.** *We say that the VRS is* binding*, if a cheating prover* $\mathtt{P}^*$ *cannot find a new opening* $\hat{d}\neq d$ *and a new commitment* $\hat{c}\neq c$ *accepted together with the proof* π*, where* (c,d,π) *was generated in a honest*

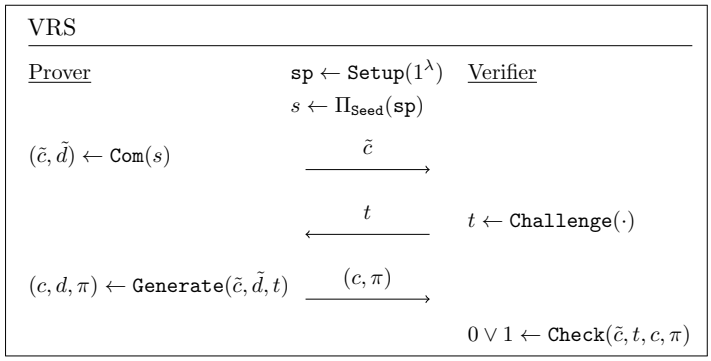

Figure 6: Our abstract verifiable random secret scheme.

run of the protocol, depending on the commitment $\tilde{c}$ and the challenge t. We define the binding advantage $\mathbf{Adv}^{\mathbf{B}}$ *of the cheating prover* $\mathtt{P}^*$ *to be:*

$$\mathbf{Adv}^{\mathbf{B}}(\mathtt{P}^*) = \Pr\left[\begin{array}{c|c} & s \leftarrow \Pi_{\mathtt{Seed}}(\mathtt{sp}) \\ c \neq \hat{c}, d \neq \hat{d} & (\tilde{c}, \tilde{d}) \leftarrow \mathtt{P}^*(\cdot) \\ \mathtt{Check}(\tilde{c}, t, c, \pi) = 1 \ : & t \leftarrow \mathtt{Challenge}(\cdot) \\ \mathtt{Check}(\tilde{c}, t, \hat{c}, \pi) = 1 & (c, d, \pi) \leftarrow \mathtt{Generate}(\tilde{c}, \tilde{d}, t) \\ & (\hat{c}, \hat{d}) \leftarrow \mathtt{P}^*(s, \tilde{c}, \tilde{d}, t, c, d, \pi) \end{array}\right].$$

Definition 24 (Prover bit-Unpredictability)**.** *We say that the VRS has* prover bit-unpredictability *if, given a balanced predicate function f of a cheating prover $\mathtt{P}^*$'s choice, the predicate $f(r)$ of the value r, where r is a function of the seed s and the challenge t, is unpredictable for $\mathtt{P}^*$. We let $\mathtt{P}^*$ choose a bit $\hat{b}$, and define the prover bit-unpredictability advantage* $\mathbf{Adv}^{\mathbf{PbU}}_{\mathbf{f}}$ *of a cheating prover $\mathtt{P}^*$, with respect to f, to be:*

$$\mathbf{Adv}^{\mathbf{PbU}}_{\mathbf{f}}(\mathtt{P}^*) = 2 \cdot |\Pr\left[\begin{array}{c|c} & s \leftarrow \Pi_{\mathtt{Seed}}(\mathtt{sp}) \\ \mathtt{Check}(\tilde{c}, t, c, \pi) = 1 \wedge f(r) = \hat{b} & (\tilde{c}, \tilde{d}, f, \hat{b}) \leftarrow \mathtt{P}^*(\cdot) \\ \vee & t \leftarrow \mathtt{Challenge}(\cdot) \\ \mathtt{Check}(\tilde{c}, t, c, \pi) = 0 \wedge \tilde{b} = 1 \ : & (c, d, r, \pi) \leftarrow \mathtt{P}^*(\tilde{c}, \tilde{d}, t, s) \\ & \tilde{b} \xleftarrow{\$} \{0, 1\} \end{array}\right] - \frac{1}{2}|.$$

Definition 25 (Verifier Secrecy)**.** *We say that the VRS has* verifier secrecy*, if a cheating verifier $\mathtt{V}^*$ is unable to distinguish between a honestly generated value r and a random r sampled from the set $\mathtt{R} = \{r(s,t) : s \leftarrow \Pi_{\mathtt{Seed}}, t \leftarrow$*

$\texttt{Challenge}\}$, *where* r *is a function of the seed* s *and the challenge* t*. We define the verifier secrecy advantage* $\mathbf{Adv}^{\mathbf{(s)VS}}$ *of a cheating verifier* $\texttt{V}^*$ *as:*

$$\mathbf{Adv}^{\mathbf{(s)VS}}(\texttt{V}^*) = 2 \cdot |\Pr\left[b = \hat{b} \ : \begin{array}{c} b \xleftarrow{\$} \{0,1\} \\ s \leftarrow \Pi_{\texttt{Seed}}(\texttt{sp}) \\ (\tilde{c}, \tilde{d}) \leftarrow \texttt{Com}(s) \\ t \leftarrow \texttt{V}^*(\tilde{c}) \\ (c, d, \pi) \leftarrow \texttt{Generate}(\tilde{c}, \tilde{d}, t) \\ r_0 \leftarrow r(s,t), r_1 \xleftarrow{\$} \texttt{R} \\ \hat{b} \leftarrow \texttt{V}^*(\tilde{c}, t, c, \pi, r_b) \end{array}\right] - \frac{1}{2}|.$$

Definition 26 (Honest-Verifier Secrecy)**.** *We say that the VRS has* honest-verifier secrecy*, if a honest but curious verifier* $\texttt{V}^*$ *is unable to distinguish between a honestly generated value* r *and a random* r *sampled from the set* $\texttt{R} = \{r(s,t) : s \leftarrow \Pi_{\texttt{Seed}}, t \leftarrow \texttt{Challenge}\}$*, where* r *is a function of the seed* s *and the challenge* t*. We define the honest-verifier secrecy advantage* $\mathbf{Adv}^{\mathbf{(s)HVS}}$ *of a honest verifier* $\texttt{V}^*$ *as:*

$$\mathbf{Adv}^{\mathbf{(s)HVS}}(\texttt{V}^*) = 2 \cdot |\Pr\left[b = \hat{b} \ : \begin{array}{c} b \xleftarrow{\$} \{0,1\} \\ s \leftarrow \Pi_{\texttt{Seed}}(\texttt{sp}) \\ (\tilde{c}, \tilde{d}) \leftarrow \texttt{Com}(s) \\ t \leftarrow \texttt{Challenge}(\cdot) \\ (c, d, \pi) \leftarrow \texttt{Generate}(\tilde{c}, \tilde{d}, t) \\ r_0 \leftarrow r(s,t), r_1 \xleftarrow{\$} \texttt{R} \\ \hat{b} \leftarrow \texttt{V}^*(\tilde{c}, t, c, \pi, r_b) \end{array}\right] - \frac{1}{2}|.$$

Remark 2. *We note a cheating verifier* $\texttt{V}^*$ *always can be turned into a honest verifier if* $\texttt{V}^*$ *has to commit to their challenge before receiving the seed commitment. This would increase the communication complexity of the protocol, but at the same time make it easier to prove security.*

Definition 27 (ϵ-Security)**.** *A VRS is said to be* ϵ*-secure, for an* ϵ *negligible in the security parameter* λ*, if the sum of all the advantages is less than* ϵ*. That is, we have* ϵ*-security if*

$$\mathbf{Adv}^{\mathbf{S}}(\texttt{P}^*) + \mathbf{Adv}_{\mathbf{f}}^{\mathbf{PbU}}(\texttt{P}^*) + \mathbf{Adv}_{\mathbf{f}}^{\mathbf{HVS}}(\texttt{V}^*) \leq \epsilon(\lambda).$$

Remark 3. *We note that the only contribution of the verifier (before verifying the proof) is to provide some randomness, and hence, the protocol is public coin. It follows that we can use the Fiat-Shamir heuristics to make*

the protocol non-interactive by letting $t = H(\mathtt{sp}, \tilde{c})$ *(also including* s *if it is public), for a hash-function* H*. In this scenario, the prover sends the transcript* $(\tilde{c}, c, \pi)$ *to the verifier, and stores* c, d *himself. The verifier then runs the* `Check` *algorithm as usual to make sure that everything is correct. However, note that in this case we lose Prover bit-Unpredictability, where* $\mathtt{P}^*$ *can try as many times as desired before sending the proof, and hence able to control a few bits of* r*. This may or may not be important for the subsequent application.*

5 Subliminal-Free Digital Signatures

5.1 How to Achieve a Subliminal-Free Channel?

Let `S` be the sender, `R` be the receiver and `W` the warden. We consider `S`, `R` and `W` to all be probabilistic polynomial time algorithms. Let $\mathcal{C}$ be an information channel controlled by `W`, allowing `S` to send information to `R`. We want to define what it means for $\mathcal{C}$ to be a subliminal-free channel, and in particular, what it means for $\mathcal{C}$ to be a subliminal-free channel when $\mathcal{C}$ is a message authentication without secrecy channel. That is, $\mathcal{C}$ is a covert-free channel where messages from `S` are sent in the clear to `R` together with a signature, so that `R` can be sure that the message indeed is from `S`. Further, `W` wants to be sure that there is no extra hidden information being sent. There are many ways to encode information into a message-signature pair, where we only are interested in preventing methods that encode information into the signature – see the Warden Model in Section 1.1.

In our model `S` can hide a subliminal message in the signature, in which `R` later can extract using some pre-shared information. Signatures can be either deterministic or probabilistic. If the signature scheme has been made deterministic by derandomization, `W` could require `S` to prove in zero-knowledge that the randomness used in the signature is deterministically generated. If the signature scheme does not use any randomness and is deterministic, `S` is not required to prove anything. In the case of probabilistic

signatures, a subliminal signer could easily choose a subliminal message (or an encryption of a subliminal message) to be the randomness used in the signature, independent of the message being sent. Hence, if probabilistic signature schemes are allowed, we must be able to restrict S's control over the choice of randomness used in the signatures—we are able to do this by using a VRS. We additionally require a proof stating that the signature is honestly generated.

Lastly, we require $\mathcal{C}$ to be a reliable channel. In some cases the honestly generated randomness used in the signature is equal to the subliminal message S wants to send. This could happen if the subliminal messages are very short, or divided into small pieces of sizes down to only 1 bit per signature. If S gets to decide if they want to send the authenticated message or not, after given the message to send, or after interaction with W, or after signature-generation, then this can be used to create a subliminal channel. For every signature, S checks if their subliminal message is embedded or not; if it is, then S sends the message, and otherwise aborts and tries again. This channel was pointed out by Desmedt [Des96]. It follows that the warden cannot allow S to abort if he wants the channel to be subliminal-free. However, if S is using a probabilistic signature scheme and is allowed to generate the signatures in a verifiable but non-interactive manner, they will be able to abort without W noticing, an aspect seemingly inherent to the construction. We will thereby give two different definitions of subliminal-free digital signatures, where the relaxed notion will allow a small subliminal channel in the construction, given that S have to work exponentially hard to achieve this.

5.2 Subliminal and Subliminal-Free Digital Signatures

We continue by more informally defining what we mean by subliminal and subliminal-free digital signature schemes.

Definition 28 (Subliminal Digital Signatures)**.** *Let m be a fixed message, $\mathcal{S}$ a signature scheme, $\hat{m}$ a subliminal message chosen by* $\mathtt{S}^*$ *and s some*

secret pre-shared information between $\mathtt{S}^*$ *and* $\mathtt{R}^*$*. Then we say that* $\mathcal{S}$ *is a subliminal digital signature scheme if* $\mathtt{S}^*$ *can encode* $\hat{m}$ *into a signature* σ *using an algorithm* $\mathtt{Encode}_{\mathtt{S}^*}$*, where* σ *is a valid signature of* m *with respect to* $\mathcal{S}$*, that can be decoded by* $\mathtt{R}^*$ *using an algorithm* $\mathtt{Decode}_{\mathtt{R}^*}$*, but cannot be decoded nor detected by* $\mathtt{W}$*. That is,*

- $\mathtt{Encode}_{\mathtt{S}^*}$ *on input the message* m*, signing key* $\mathtt{sk}$*, and subliminal message* $\hat{m}$*, outputs a signature* σ *of the message* m *and a proof* π*,*
- $\mathtt{Decode}_{\mathtt{R}^*}$ *on input a message* m*, a signature* σ*, and secret pre-shared information* s*, outputs the subliminal message* $\hat{m}$*.*

Definition 29 (Subliminal-Free Digital Signatures)**.** *Let* m *be a fixed message,* $\mathcal{S}$ *a signature scheme,* $\hat{m}$ *a subliminal message chosen by* $\mathtt{S}$ *and* s *some secret pre-shared information between* $\mathtt{S}$ *and* $\mathtt{R}$*. Further, let* σ *be a valid signature of* m *with respect to* $\mathcal{S}$ *and let* π *be a proof of correctness, both potentially generated by interaction between* $\mathtt{S}$ *and* $\mathtt{W}$*. Assume that* $\mathtt{W}$ *will close the channel if* $\mathtt{S}$ *deviates from the protocol. Then we say that* $\mathcal{S}$ *is a subliminal-free digital signature scheme if* $\mathtt{S}$ *cannot reliably both create a proof* π *accepted by* $\mathtt{W}$ *and at the same time encode a subliminal message* $\hat{m}$ *into* σ *that can be decoded by* $\mathtt{R}$ *but cannot be decoded nor detected by* $\mathtt{W}$*.*

However, as noted earlier, if $\mathcal{S}$ is a probabilistic signature scheme used in the non-interactive setting, then $\mathtt{S}$ can undetectably abort the protocol after generating a signature on m and restart the signing algorithm. We therefore additionally include a relaxed notion of a subliminal-free digital signature scheme, where we allow a small subliminal channel.

Definition 30 (Subliminal l-Free Digital Signatures)**.** *Let* m *be a fixed message,* $\mathcal{S}$ *a signature scheme,* $\hat{m}$ *a subliminal message chosen by* $\mathtt{S}$ *and* s *some secret pre-shared information between* $\mathtt{S}$ *and* $\mathtt{R}$*. Further, let* σ *be a valid signature of* m *with respect to* $\mathcal{S}$ *and let* π *be a proof of correctness. Assume that* $\mathtt{W}$ *will close the channel if* $\mathtt{S}$ *deviates from the protocol. Then we say that* $\mathcal{S}$ *is a subliminal* l*-free digital signature scheme if* $\mathtt{S}$ *cannot reliably both create a proof* π *accepted by* $\mathtt{W}$ *and at the same time encode a subliminal message* $\hat{m}$ *of size greater or equal to* l *into* σ *decodable by* $\mathtt{R}$*, but*

cannot be decoded nor detected by W *unless* S *does work exponentially in* l.

5.3 Subliminal-Free Digital Signature Scheme

Deterministic signatures seem to have an advantage over probabilistic signatures as they can guarantee a subliminal-free digital signature scheme in the non-interactive setting. However, they are usually harder to construct and even harder to prove correctly evaluated. Probabilistic signatures schemes are used more in practice, and we want to extend our definition of subliminal-free digital signature schemes to allow interaction between S and W to ensure that we can achieve the strongest security guarantees also for these schemes.

Definition 31 (Subliminal-Free Digital Signature Scheme)**.** *A subliminal-free digital signature scheme consists of an interactive signing protocol* ($\Pi_{\texttt{Sign}}$) *and five algorithms: key generation* (KeyGen)*, setup* (Setup)*, verification* (Verify)*, and checking* (Check)*, where*

- KeyGen*, on input the security parameter* 1^λ*, outputs public parameters* pp*, a signing key* sk*, and a verification key* vk*,*
- Setup*, on input security parameter* 1^λ*, outputs public parameters* sp*,*
- $\Pi_{\texttt{Sign}}$*, on input message* m *and* sk*, outputs signature* σ *and proof* π*,*
- Verify*, on input* m*,* σ *and* vk*, outputs either* 0 *or* 1*,*
- Check*, on input* m*,* σ*,* vk *and* π*, outputs either* 0 *or* 1*,*

and the public parameters pp *are implicit inputs to all algorithms following* KeyGen *and the public setup parameters* sp *are implicit inputs to* Sign *and* Check*. We require that* Check *returns* 1 *if and only if* Verify *returns* 1 *and* π *is valid.*

In our constructions (see Section 6) we use an interactive VRS as a part of the signature protocol to ensure that the randomness is honestly generated, and give a proof for this statement. The checking algorithm is then the

checking algorithm of the VRS combined with the verification algorithm of the signature scheme.

Remark 4. *Note that there are some small differences in how we define subliminal-free digital signatures compared to how it is defined by Bohli et al. [BGVS07]. They require it to be exponentially hard in the security parameter to find two different pairs of signatures and proofs that are valid for the same message. This is essentially the same as in our case when the signature scheme is deterministic, but we also allow for a probabilistic signature scheme. In addition, they have a* `SFKeyGen` *algorithm generating the checking information for the warden. We have decided to include an algorithm* `Setup`*. The setup algorithm generates publicly available information used by the sender to give a proof of correct signing, and is used by the warden to verify this proof. The proof itself should only be available to the warden, as it can contain subliminal information that can be decoded if shared with the receiver, but the setup parameters could be available to anyone.*

5.4 Subliminal-Free Digital Signatures with Pre-Processing

The subliminal-free digital signature scheme requires `S` and `W` to interact every time `S` wants to sign a message. We can lower the communication complexity of our scheme by allowing some pre-processing between `S` and `W`. In this case we make the `Setup` algorithm into an interactive protocol $\Pi_{\texttt{Setup}}$, where `S` and `W` use the interactive VRS to generate lots of commitments to randomness to be used in the signatures, together with proofs that the randomness is honestly generated. Then the commitments and the randomness are inputs to the non-interactive signature scheme. As the randomness is independent of the message to be signed, this allows us to move all communication from the signature procedure to the setup procedure, which can be executed before any messages need to be sent over the channel.

Note that in this case we get a stateful digital signature scheme, as the warden must ensure that the randomness is used in the correct order.

Otherwise, this would open a subliminal channel where the signer can choose the randomness from a set of values dependent on the subliminal messages to send.

Remark 5. *Note that in this case the setup parameters* $\mathtt{sp}$ *are still public, while the output of the VRS is not.*

5.5 Non-Interactive Subliminal-Free Digital Signatures

We can make a trade-off between communication and soundness by changing the interactive subliminal-free digital signature scheme into a non-interactive scheme by letting all the parts of the scheme be non-interactive algorithms. This would also enforce the VRS to be a non-interactive algorithm, as a part of the signing algorithm. We know from earlier that a non-interactive VRS is not bit-unpredictable. In this case we get a l-subliminal channel, as the sender can run the non-interactive VRS until they get a l-bit subliminal message encoded into the randomness used in the signature, and hence, we have a l-subliminal signature scheme.

5.6 Security of Subliminal-Free Digital Signatures

We require that the subliminal-free digital signature scheme has the same security as a normal signature scheme, plus the additional requirement that a cheating prover cannot decide the randomness used in the signature, and that a cheating verifier can't use the proof of correctness to forge signatures. Hence, we want the subliminal-free digital signature scheme to be *complete*, *sound* and secure against *existential forgery.*

Definition 32 (Completeness)**.** *We say that a subliminal-free digital signature scheme is* complete *if honestly generated pairs of signatures and proofs are accepted by both the verification and checking algorithms. Hence, we want that*

$$\Pr\left[\begin{matrix}\mathtt{Verify}(m,\sigma,\mathtt{vk})=1\\ \mathtt{Check}(m,\sigma,\mathtt{vk},\pi)=1\end{matrix} : \begin{matrix}(\mathtt{pp},\mathtt{sk},\mathtt{vk})\leftarrow\mathtt{KeyGen}(1^\lambda)\\ \mathtt{sp}\leftarrow\mathtt{Setup}(1^\lambda)\\ (\sigma,\pi)\leftarrow\mathtt{Sign}(m,\mathtt{sk})\end{matrix}\right]=1.$$

Definition 33 (Soundness)**.** *A subliminal-free signature scheme is* sound *if the sender is unable to establish a* subliminal bit channel *with the receiver. A subliminal bit channel is a channel where a subliminal signer* $\mathtt{S}^*$*, given a message* m*, can* reliably *encode a* subliminal bit $\hat{m} \in \{0,1\}$ *into a valid signature* σ *of* m *that the subliminal receiver* $\mathtt{R}^*$ *can decode. We also require* $\mathtt{S}^*$ *to produce a proof* π *of correctness accepted by the warden* $\mathtt{W}$*. That is, the subliminal-free signature scheme is sound if*

$$\left|\Pr\left[\begin{array}{c}\mathtt{Decode}_{\mathtt{R}^*}(m,\sigma,s)=\hat{m}\\ \mathtt{Verify}(m,\sigma,\mathtt{vk})=1\\ \mathtt{Check}(m,\sigma,\mathtt{vk},\pi)=1\end{array} : \begin{array}{c}\hat{m}\xleftarrow{\$}\{0,1\}\\ (\mathtt{pp},\mathtt{sk},\mathtt{vk})\leftarrow\mathtt{KeyGen}(1^\lambda)\\ \mathtt{sp}\leftarrow\mathtt{Setup}(1^\lambda)\\ (\sigma,\pi)\leftarrow\mathtt{Encode}_{\mathtt{S}^*}(m,\mathtt{sk},\hat{m})\end{array}\right]-\frac{1}{2}\right|<\epsilon(\lambda),$$

where the algorithms $\mathtt{Encode}_{\mathtt{S}^*}$ *and* $\mathtt{Decode}_{\mathtt{R}^*}$ *are as in Definition 28.*

A subliminal bit-channel is the simplest subliminal channel, where the sender wants to send a signal, e.g., "escape tonight." With 50 % probability, the first bit of a signature is equal to the subliminal bit chosen by the sender, that is, by chance they are able to send a subliminal bit. However, if the received bit is the intended subliminal message with 50 % probability then it cannot be trustworthy. The receiver cannot gamble and perform a predetermined, possibly daring, action based on a random bit. This cannot be a subliminal bit channel.

Definition 34 (Existential unforgeability)**.** *We say that a subliminal-free digital signature scheme is* existential unforgeable *if an adversary* $\mathtt{A}$*, after given valid signatures* σ_i *and proofs* π_i *of messages* m_i *of* $\mathtt{A}$*'s choice from a signing oracle* $\mathtt{O}_{\mathtt{sign}}$*, cannot forge a signature on any new message under the same public-private key pair. We define the existential unforgeability advantage* $\mathbf{Adv}^{\mathbf{EUF}}$ *of an adversary* $\mathtt{A}$ *to be:*

$$\mathbf{Adv}^{\mathbf{EUF}}(\mathtt{A})=\Pr\left[\begin{array}{c}\hat{m}\notin\{m_i\}\\ \mathtt{Verify}(\hat{m},\hat{\sigma},\mathtt{vk})=1\end{array} : \begin{array}{c}(\mathtt{pp},\mathtt{vk})\leftarrow\mathtt{KeyGen}(1^\lambda)\\ \mathtt{sp}\leftarrow\mathtt{Setup}(1^\lambda)\\ \{(\sigma_i,\pi_i)\}\leftarrow\mathtt{A}^{\mathtt{O}_{\mathtt{sign}}}(\mathtt{pp},\mathtt{vk},\{m_i\})\\ (\hat{m},\hat{\sigma})\leftarrow\mathtt{A}(\{(m_i,\sigma_i,\pi_i)\}_i)\end{array}\right]\leq\epsilon(\lambda).$$

6 Our Schemes

We present two VRS schemes: one based on lattices and one based purely on one-way functions. Then we combine both of them with lattice-based

signatures, and we analyze the size and computation of both schemes.

6.1 A Lattice-Based VRS from Shuffled Randomness

Our lattice-based VRS scheme is built on top of the commitment scheme by Baum et al. [BDL$^+$18] combined with the verifiable shuffle of known content by Silde et al. [ABG$^+$, BGS]. Let `P` denote the prover and `V` the verifier.

The goal of the VRS is to produce elements from R_p, where the coefficients are Gaussian distributed with standard deviation σ. We need three such elements to produce a signature. The protocol works as follows:

1. `Seed`: `V` draws τ Gaussian distributed polynomials s_i from R_p with standard deviation $\sigma/\sqrt{\kappa}$ and sends them to `P`.
2. `Commit`: `P` shuffles the polynomials using a random permutation γ, commits to them in the new order, and sends the commitments to `V`.
3. `Challenge`: `V` draws three random subset T_j, for $1 \leq j \leq 3$, each of size κ, of indices from 1 to τ and sends them to `P`.
4. `Generate`: `P` sums together the commitments for each set of indices, and sends the sums to `V` together with the proof of shuffle.
5. `Check`: `V` verifies that the sums and the proof of shuffle are correct.

The VRS scheme is visualized in Figure 7. Note that we can sum together the commitments as long as $\kappa \leq 4\sigma$, which is the criteria for accepting an opening of a commitment. If κ is too large, we can sum together 4σ commitments at a time, and then give a proof of linearity in the end. We proceed by proving *Completeness*, *Binding*, *Prover bit-Unpredictability* and *Honest-Verifier Secrecy* for the lattice-based VRS.

Lemma 1 (Completeness)**.** *The Lattice-VRS in Figure 7 is complete.*

Proof. Assuming that the prover is honest, he will commit to the given seed-values in a permuted order. He will also be able to generate an accepting

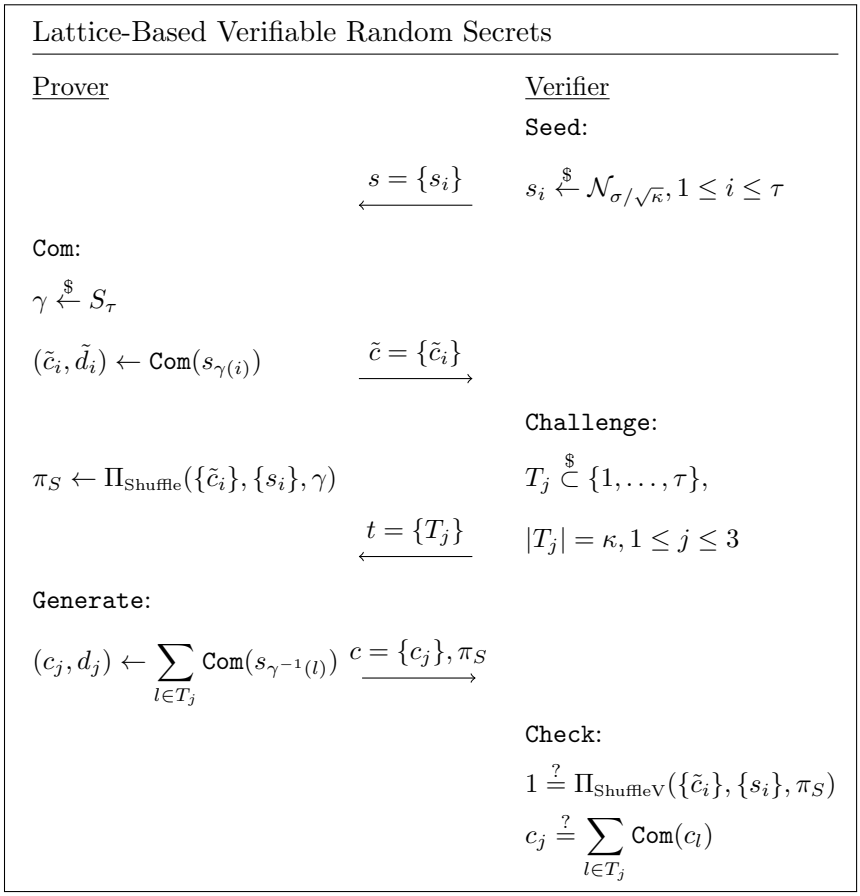

Figure 7: The lattice-based verifiable random secret scheme, using the commitment scheme by Baum et al. [BDL+18] and the verifiable shuffle of known content by Silde et al. [ABG+, BGS].

zero-knowledge proof of correct shuffle, as the shuffle-protocol is complete. Further, if $\kappa \leq 4\sigma$, then we can sum together the commitments directly, and if $\kappa > 4\sigma$ we create a new commitment to the sum of the underlying messages and give a zero-knowledge proof of the correct sum. Either way the sum is correct, as the sum-protocol is complete. We conclude that the VRS is complete. □

Lemma 2 (Binding)**.** *The Lattice-VRS in Figure 7 is binding.*

Proof. Assuming that a cheating prover $\mathtt{P}^*$ has advantage ϵ of breaking the binding property of the VRS, then $\mathtt{P}^*$ also has the advantage ϵ of breaking the binding property of the commitment scheme. The binding property of the commitment scheme is based on the R-SIS-problem, and hence, an attacker $\mathtt{A}$ can then turn $\mathtt{P}^*$ into a R-SIS-solver with success probability ϵ. We conclude that the VRS is binding. □

Lemma 3 (Prover bit-Unpredictability)**.** *The Lattice-VRS in Figure 7 is prover bit-unpredictable.*

Proof. The cheating prover $\mathtt{P}^*$ wants to predict a bit $\hat{b}$ of $r_j = r(\{s_l\}_{l=1}^{\kappa}, \{\hat{c}_l\}_{l=1}^{\kappa}, T_j)$, for $0 \leq j \leq 3$, that is, a bit $\hat{b}$ of the values $r_j = \sum_{l \in T_j} s_{\gamma^{-1}(l)}$ for the permutation γ of the underlying messages of the set of commitments. All values s_i are drawn at random from a Gaussian distribution. $\mathtt{P}^*$ is allowed to guess after they know all s_i's, but only before receiving the sets T_j of indices. As each T_j are random sets of indices drawn uniformly at random, then r_j is a random sum of κ random elements, and hence, r_j is random. The probability of guessing any bits if r_j correctly is $1/2$, and hence, the prover bit-unpredictability advantage $\mathbf{Adv}_{\mathbf{f}}^{\mathbf{PbU}}(\mathtt{P}^*)$ is negligible. We conclude that the VRS is prover bit-unpredictable. □

Lemma 4 (Honest-Verifier Secrecy)**.** *The Lattice-VRS in Figure 7 has honest-verifier secrecy.*

Proof. Assume that the commitments are hiding. Further, assume that a honest but curious verifier $\mathtt{V}^*$ is given the value r_b at the end of the

protocol. Their task is to decide if r_b is a sum of κ elements among the values $s_1, \ldots, s_\tau$ or not. This reduces to the k-SUM problem, for a subset of size κ out of τ elements. If $\mathtt{V}^*$ can break the honest-verifier secrecy of the VRS with a probability ϵ, then an attacker $\mathtt{A}$ can then turn $\mathtt{V}^*$ into a k-SUM solver with success probability ϵ. We conclude that the VRS has honest-verifier secrecy. □

Theorem 1. *The lattice-based VRS detailed in Figure 7 is complete and ϵ-Secure.*

Proof. This follows from combining Lemma 1, 2, 3 and 4. □

6.2 A Schnorr-Like SFS from the Lattice-Based VRS

The lattice-based VRS in the previous section give us three commitments c_1, c_2, c_3 that are known to the warden $\mathtt{W}$, and valid openings, $(y_1, \boldsymbol{r}_1), (y_2, \boldsymbol{r}_2)$, $(y_3, \boldsymbol{r}_3)$ known to the sender $\mathtt{S}$. All the y_i's are Gaussian distributed elements from R_p with standard deviation σ. Finally, $\mathtt{S}$ computes $\langle \boldsymbol{y}, \boldsymbol{a} \rangle = y_1 + y_2 a_1 + y_3 a_2 = t'$, where t' is the committed randomness used in the signature. $\mathtt{S}$ attach a proof of linearity for this statement, and complete the signature given a message m. Finally, $\mathtt{S}$ sends the message m, the signature $(t', \boldsymbol{z})$ together with the output of the VRS to $\mathtt{W}$, which then verifies that the proofs and signature are correct. If, and only if, both the `Check-` and `Verify` algorithms outputs 1, $\mathtt{W}$ forwards the message and signature to the receiver $\mathtt{R}$. The full subliminal-free signature scheme is presented in Figure 8.

Theorem 2. *The Lattice-SFS scheme in Figure 8 is complete, sound and secure against existential forgeability.*

Proof. Assume that the lattice-based VRS is complete and ϵ-secure and that the signature scheme is complete and secure against existential forgability.

Completeness: Follows directly from the fact that the VRS and the signature schemes are complete.

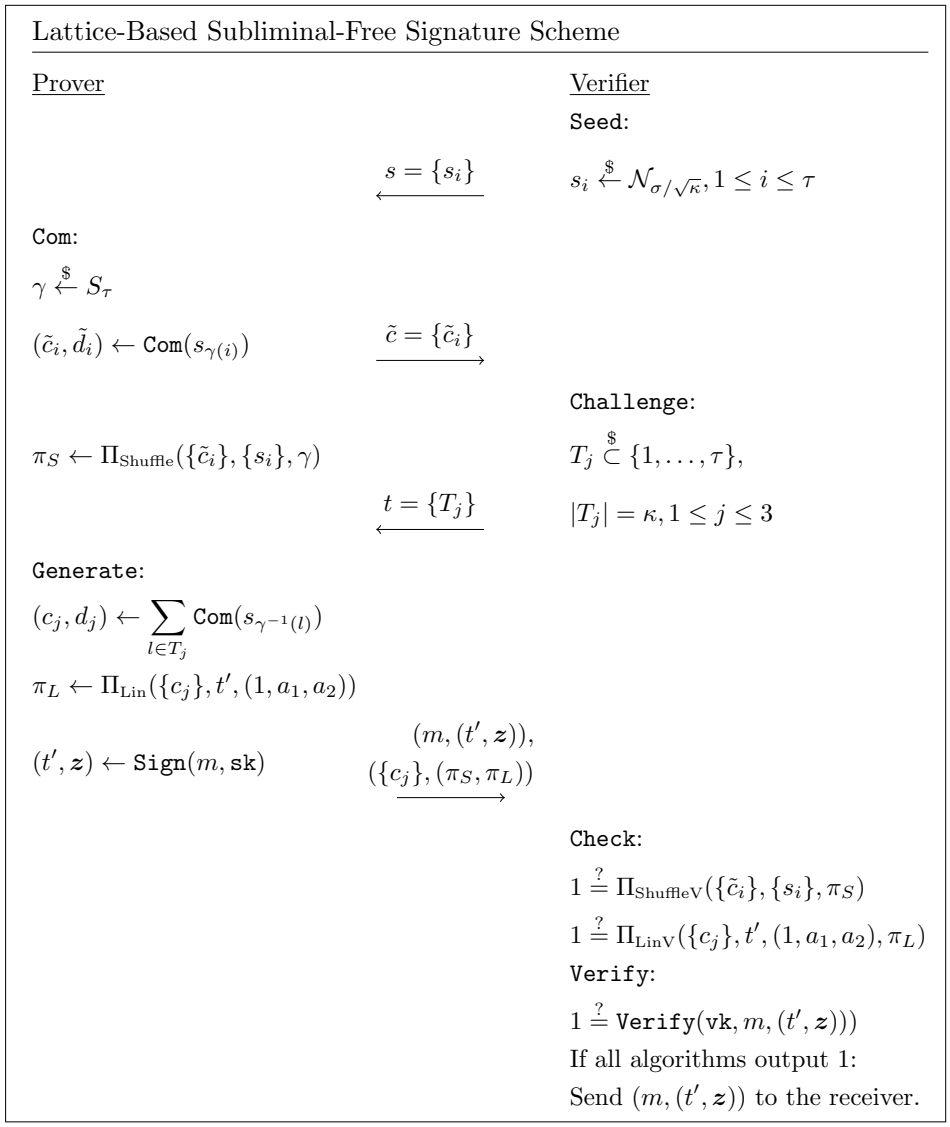

Figure 8: The lattice-based subliminal-free digital signature scheme, using the lattice-based VRS and a lattice signature scheme based on Lyubashevsky [Lyu09, Lyu12].

Soundness: The VRS ensures that the randomness is honestly generated, and because of the prover bit-unpredictability of the VRS, a cheating prover $\mathtt{P}^*$ has a negligible probability of embedding any information into the randomness used in the signature. It follows that a cheating prover doesn't have a reliable subliminal channel.

Existential forgeability: A honest but curious verifier $\mathtt{V}^*$ learns $A\boldsymbol{r}$ for some public uniformly random matrix A and short randomness $\boldsymbol{r}$, where $\boldsymbol{r}$ is the output of the VRS. By multiplying all possible 3-subsets $\boldsymbol{s}_j$ of $s_1, \ldots, s_\tau$ by A, this reduces to solving the k-SUM problem for $k = \kappa$ when the set is $\{A\boldsymbol{s}_j\}_{i=j}^{\tau}$. This search problem is equivalent to the decision problem with the same input. The VRS ensures that the randomness is honestly generated, and because of the honest verifier secrecy of the VRS, $\mathtt{V}^*$ has a negligible probability of learning any information about the secret signing key used to generate the signature. Furthermore, both zero-knowledge proofs in the VRS can be simulated, and hence, this additional information does not provide any information that $\mathtt{V}^*$ can use to win the forgeability-game.

We conclude that the lattice-based SFS detailed in Figure 8 is complete, sound and secure against existential forgeability. □

6.3 Generic VRS Framework Based on One-Way Functions

Let $\mathtt{H}$ be a cryptographic hash-function with output length 2λ, and let $\mathtt{OWF}$ a one-way function used in any "hash-then-sign" signature scheme. We will now build a VRS using only $\mathtt{H}$ and $\mathtt{OWF}$. The VRS is using a "cut-and-choose" technique similar to what is used in the preprocessing phase from the MPC scheme by Katz et al. [KKW18] and in the zero-knowledge proofs and signature schemes by Beullens [Beu20].

Let s be a binary seed of length 2λ, chosen by the prover. The value s will be the root of a binary tree. The two children h_0 and h_1 of s will have the values $h_0 = \mathtt{H}(s, 1)$ and $h_1 = \mathtt{H}(s, 2)$. Further, the children of h_0 will have the values $h_{00} = \mathtt{H}(h_0, 1), h_{01} = \mathtt{H}(h_0, 2)$ and the children of h_1 will have

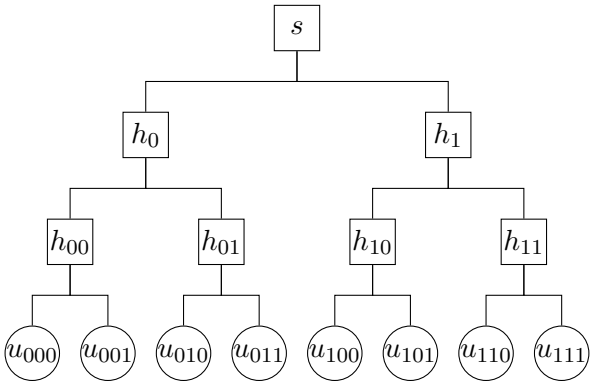

Figure 9: The binary tree generated by seed s using hash function $\mathtt{H}$.

the values $h_{10} = \mathtt{H}(h_1, 1), h_{11} = \mathtt{H}(h_1, 2)$, and so on. This is illustrated in Figure 9. For a predefined height h, the tree will have $M = 2^h$ leaf nodes, denoted $u_0, ..., u_{M-1}$. As each node is the output of the hash function, each value u_i will be a uniformly random string of length 2λ.

In the case that the subsequent application, e.g. a subliminal-free "hash-then-sign" signature scheme, needs uniformly distributed randomness, then no further steps are needed. If we need some special distribution, then we use the nodes $u_0, ..., u_{M-1}$ as input to an algorithm that produces the correct distribution. If we want to produce similar lattice-based signatures as from the previous constriction, we can evaluate the algorithm $\mathtt{GaussSampler}_\sigma$ on the leaf nodes to produce the desired outcome. Let's denote the new values by $v_0, ..., v_{M-1}$, independent of whether they changed or not.

Then we apply the one-way function $\mathtt{OWF}$ on the values on $v_0, ..., v_{M-1}$ to produce $w_0, ..., w_{M-1}$, where $w_i = \mathtt{OWF}(v_i)$ for $0 \leq i \leq M-1$. For the lattice-based signatures we would have $w_i = \mathtt{OWF}(v_i) = Av_i$, where v_i is a vector of Gaussian distributed elements from R_p and A is a public matrix.

Then we compute a Merkle-tree, from the bottom up. The hash $\mathtt{H}(w_0, w_1)$ gives the parent of w_0 and w_1, and so on. Denote the root node of the

Merkle-tree by $\tilde{c}_s$. Then $\tilde{c}_s$ is a commitment to the full tree, computed deterministically from s. The prover sends $\tilde{c}_s$ to the verifier. Then the verifier responds with a single index $t = I$, where $0 \leq I \leq M - 1$ is chosen uniformly at random. The output of the VRS in then the secret value v_I, only known to the prover, together with the public commitment $c = w_I = \texttt{OWF}(v_I)$ and a proof π_I of correctness.

The proof π_I is generated in the following way. We want to prove that the full tree is generated correctly and that the commitment w_I is the Ith leaf node, but at the same time preserve the privacy of v_I. If we give away s, the verifier can re-compute the full tree to verify that $\tilde{c}$ is correct, but will also learn v_I. This means that s must stay private. However, we can publish all intermediate values in the tree that is not on the path from s to u_I, and to minimize communication, it is enough to send the roots of the subtrees not in that path. Then the verifier can compute the remaining values themselves.

As an example, assume that $h = 3$ as in Figure 9, and let $I = 011$. Then $\pi_I = (h_1, h_{00}, u_{010})$, and the verifier can compute the values u_{000} and u_{001} from h_{00} and u_{100} u_{101}, u_{110} and u_{111} from h_1. Given all the leaf nodes, except u_{011}, the verifier can compute the values $v_{000}, ..., v_{010}, v_{100}, ..., v_{111}$ and $w_{000}, ..., w_{010}, w_{100}, ..., w_{111}$. Finally, with the knowledge of w_{011}, one can re-compute the Merkle-tree and the root $\tilde{c}$, and check that it is the same as that given in the commitment phase.

The full VRS protocol is illustrated in Figure 10 and it works as follows:

1. `Seed`: `P` chose a random bit string s of length 2λ and keeps this private.
2. `Commit`: `P` generates the full tree applying the algorithm `BuildTree` on s, and sends the root $\tilde{c}$ to `V` as a commitment.
3. `Challenge`: `V` draws a random index $t = I$, where $0 \leq I \leq M - 1$, and sends t to `P`.
4. `Generate`: `P` publishes $c = w_I$ and the proof π_I, generated by applying the algorithm `SubTrees` on s and I, which contains the roots of the

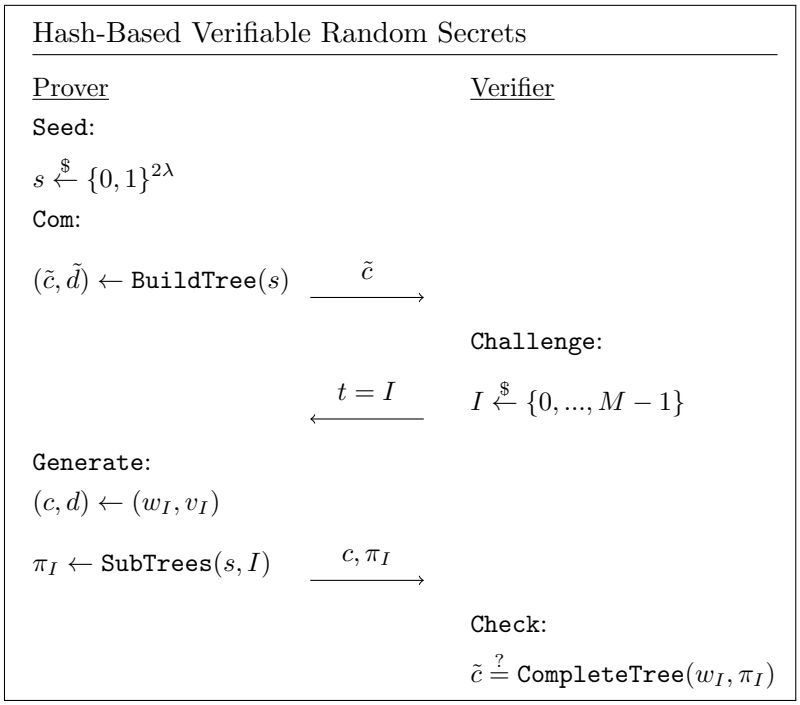

Figure 10: The hash-based verifiable random secret scheme, using the underlying cryptographic hash function `H` and the one-way function `OWF`.

subtrees not on the path between s and u_I.

5. `Check`: `V` verifies that w_I and π_I generates the tree by applying the algorithm `CompleteTree` to w_I and π_I and comparing the root to $\tilde{c}$.

We note that the security properties of this VRS follows directly from the properties of cryptographic hash functions and Merkle-trees.

Theorem 3. *The hash-based VRS detailed in Figure 10 is complete and ϵ-Secure.*

Proof. Assume that `H` and `OWF` are cryptographically sound one-way func-

tions each ensuring λ bits of security and that $M = 2^{\lambda}$.

Completeness: Follows directly from the completeness of a Merkle-tree, where the full tree is generated deterministically from the seed s.

Binding: Follows directly from the collision resistance of `H` and `OWF`.

Prover bit-Unpredictability: Follows from the fact that the cheating prover has probability $1/M$ to guess the correct index I before committing to $\tilde{c}$. If the guess is correct, they can interchange the correct value w_I with another value $\hat{w}_I$ of their own choice when building the tree, and the proof will go through. If the wrong index is guessed, the cheating prover will be caught by the verifier. This is unless they break the collision resistance of the hash function, as the root node will be different.

Honest-Verifier Secrecy: Follows directly from the fact that the outputs of `H` and `OWF` are indistinguishable from uniformly random.

We conclude that the hash-based VRS detailed in Figure 10 is complete and ϵ-Secure. □

6.4 A Hash-Then-Sign SFS from One-Way Functions

Let $\mathcal{S}$ be a "hash-then-sign"-signature scheme producing signatures of the form (w, z). Then, combining the hash-based VRS from the previous subsection with $\mathcal{S}$, we get a subliminal-free digital signature scheme. The scheme is presented in Figure 11.

Theorem 4. *The SFS scheme in Figure 11 is complete, sound and secure against existential forgability.*

Proof. Assume that the hash-based VRS is complete and ϵ-secure and that the signature scheme is complete and secure against existential forgability.

Completeness: Follows directly from the fact that the VRS and the signature schemes are complete.

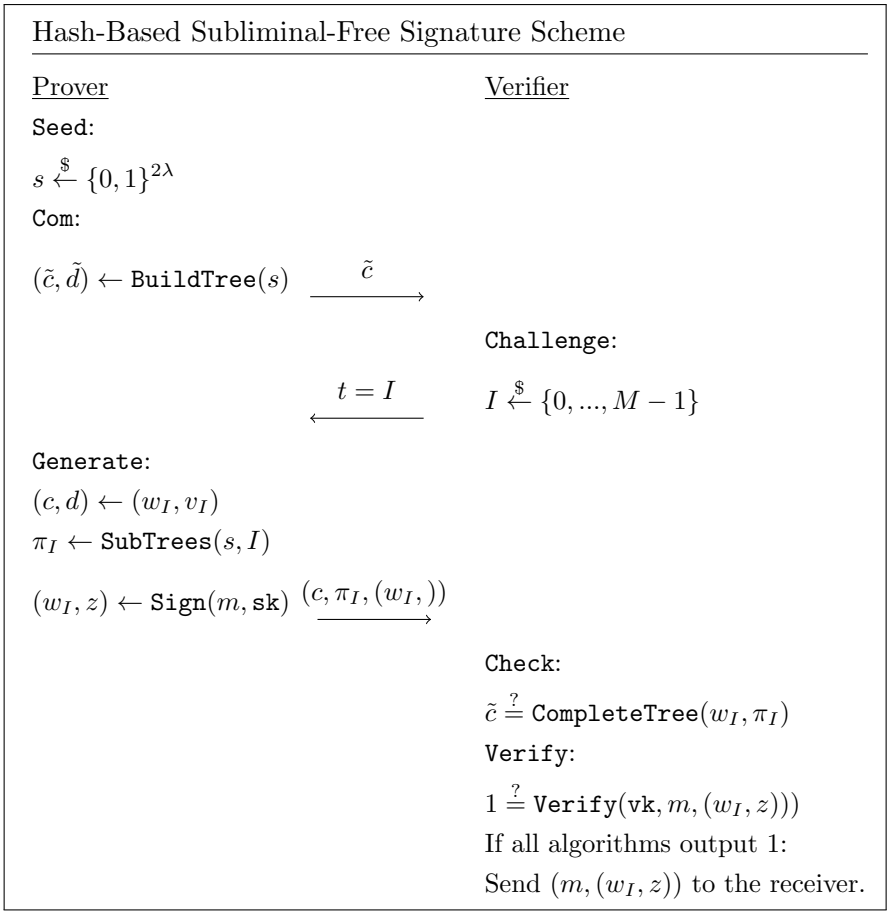

Figure 11: The hash-based subliminal-free digital signature scheme, using the hash-based VRS and a "hash-then-sign" signature scheme.

Soundness: The VRS ensures that the randomness is honestly generated, and because of the prover bit-unpredictability of the VRS, a cheating prover $\mathtt{P}^*$ has a negligible probability of embedding any information into the randomness used in the signature. It follows that a cheating prover doesn't have a reliable subliminal channel.

Existential forgeability: A honest but curious verifier $\mathtt{V}^*$ learns $w_I = \mathtt{OWF}(v_I)$ for some public one-way function $\mathtt{OWF}$ and secret randomness v_I; the outputs from the VRS. The VRS ensures that the randomness is honestly generated, and because of the honest verifier secrecy of the VRS, $\mathtt{V}^*$ has a negligible probability of learning any information about the secret signing key used to generate the signature, and hence, this additional information does not provide any information that $\mathtt{V}^*$ can use to win the forgeability-game.

We conclude that the hash-based SFS detailed in Figure 11 is complete, sound and secure against existential forgeability. □

6.5 Efficiency and Size

The lattice-based SFS. Here we run the interactive VRS, generate proofs and produce a signature. The sender sends τ commitments (assuming that κ is small enough), a shuffle proof, a linearity proof, and a message-signature pair. Warden sends τ Gaussian distributed values and three challenge sets. The warden's messages can be replaced with short seeds, saving lots of communication. In addition, letting the sender choose their own seeds gives us a non-interactive SFS with a l-bit subliminal channel, with a huge overhead for the sender. Set the security parameter $\lambda = 128$ bits.

To generate seeds $\{s_i\}_{i=1}^{\tau}$, we can use the deterministic Gaussian sampler from Algorithm 11 in qTesla [AAB+19], denoted $\mathtt{GaussSampler}_\sigma$. Choosing a binary string ρ of length 2λ, the values s_i can be generated as $s_i \leftarrow \mathtt{GaussSampler}_\sigma(\rho, i)$. Hence, we only have to send ρ, and the verifier can check that all values were computed correctly when verifying the proof of

shuffle. Furthermore, the sets T_j can be generated as hashes $\mathtt{H}(j, \hat{\rho}, \{\tilde{c}_i\}_{i=1}^{\tau}, l)$ for $1 \leq j \leq 3$, $1 \leq l \leq \kappa$, and a suitable hash function $\mathtt{H}$. Here, $\hat{\rho} = \rho$ in the non-interactive case, while $\hat{\rho}$ is a random string of size 2λ in the interactive setting. Note that if we move to the pre-processing setting, all seeds $(\rho_l, \hat{\rho}_l)$ – where $\hat{\rho}_l$ is independent of ρ_l – can be sent at once. Then the warden only needs to receive a signature, verify, and then forward it for each message being sent later on.

To achieve $\lambda = 128$ bit security, we need to decide on τ and κ such that τ choose $\kappa \geq 2^{128}$ to prevent brute-force attacks, and $\tau^{\kappa/2} \geq 2^{128}$ to resist the best algorithms to solve the k-SUM problem [Eri95]. Setting $\tau = 256$ and $\kappa = 32$, we satisfy both of these equations. We note that κ is small enough to sum the commitments directly for all practical parameters σ.

We can represent π_L as non-interactive proof $(\beta, \boldsymbol{z}, \boldsymbol{z'}, \boldsymbol{z''}, \boldsymbol{z'''})$ where β is a hash of the t's, which can later be recomputed by the verifier. Also, we note that π_S contains one commitment D_i, one ring-value s_i and a proof of linearity π_{L_i} for each $1 \leq i \leq \tau$, where each π_{L_i} is of the form $(\beta, \boldsymbol{z}, \boldsymbol{z'})$. A normal message-signature pair contains five elements from R_p. A subliminal-free signature requires an additional 2τ elements for $\{\tilde{c}_i\}_{i=1}^{\tau}$, 13 elements for π_L and 10τ elements for π_S. This sums to $12\tau + 13$ elements in R_p.

Using the optimal parameters provided by Baum et al. [BDL$^+$18] for the commitment scheme we have $p \approx 2^{32}$, $N = 1024$ and $\sigma \approx 27000$. One element in R_p is then of size $N \log p \approx 4.1$ KB, and hence, a normal signature is ≈ 16.4 KB and a subliminal-free signature is ≈ 12.65 MB. In [ABG$^+$] they estimate that it takes 1τ ms $= 0.256$ s to create the commitments, 35τ ms ≈ 9 s to generate the shuffle proof and 1.9τ ms ≈ 0.5 s to verify a shuffle proof. In summary, it takes approximately 10 seconds to generate subliminal-free signatures, and each signature is of size 13 MB if we want $\lambda = 128$ bit security. We can replace the signature scheme with more efficient lattice-based signature schemes like Dilithium [DKL$^+$] or qTesla [ABB$^+$20], where both schemes produce signatures of size 2.7 KB. This lowers the amount of data sent to the receiver by roughly 14 KB per signature. This, however, doesn't really make a difference for the sender and warden due to

the large overhead from the VRS.

The hash-based SFS. This VRS construction works with any "hash-then-sign" signature scheme, but we'll instantiate the signatures using the lattice-based signature schemes Dilithium [DKL$^+$] and qTesla [ABB$^+$20] as above. Then we achieve a post-quantum secure subliminal-free signature scheme, and can more easily compare with the previous construction based purely on lattices.

We start by observing that the computation required to prove and verify the hash-based VRS is linear in the residue of the bit-unpredictability probability of the prover, which is equivalent to the soundness of the subliminal-free signature scheme. This means that if we want soundness $1/M$ for the SFS scheme, then we need to compute $O(M)$ hashes when generating and verifying the VRS. However, we can allow the VRS to be $\epsilon(\lambda)$-secure when it comes to binding and honest-verifier secrecy to ensure security for the underlying signature scheme. But, we allow for a higher prover bit-unpredictability security, and hence larger soundness in the subliminal-free signature scheme. This way we can construct a very efficient VRS for an interactive protocol with soundness, say, one in a thousand or one in a million. This construction is not necessarily secure in the non-interactive setting, as a cheating prover can perform huge amounts of offline computations to create VRS-outputs of his choice and re-trying whenever he fails, as the VRS computations must be somewhat efficient to be used in practical situations.

To generate a binary tree of height h based on a seed s, we need to compute roughly $2 \cdot 2^h = 2M$ hashes. We do this twice in our construction, but the output nodes from the first tree are the input nodes for the second tree, and hence, we compute roughly $3M$ hashes. Also, for each leaf node u_i in the tree, we might do a computation to sample specially distributed randomness v_i, and for each v_i, we compute $w_i = \texttt{OWF}(v_i)$. Let $|\texttt{H}|$, $|\texttt{Sample}|$ and $|\texttt{OWF}|$ denote the cost it takes to compute a hash, the sampling and the one-way function, respectively. The total computation is then $T = 3M|\texttt{H}|+M(|\texttt{Sample}|+|\texttt{OWF}|)$. The verification is somewhat cheaper.

The size of the communication is relatively small. The commitment $\tilde{c}$ is the output of the hash function `H` of size 2λ, the commitment c is the output from `OWF`, and the proof π_I contains $\log M$ hash values of total size $2\lambda \log M = 2\lambda h$ bits.

We chose $\lambda = 128$ bits of security for our signature scheme, but allow for a soundness error $1/M$ where $M_1 = 2^{10}$, $M_1 = 2^{15}$ or $M_3 = 2^{20}$ in our SFS. We chose the `SHA-256` hash function. By running the command `openssl speed sha256` in the terminal, we estimate that we can compute roughly 2^{21} `SHA-256`-hashes per second with messages of size 512 bits as input. We also assume that we can compute the lattice-based commitments in roughly 1 ms, as above, which includes both `OWF` and `Sample`. qTelsa give the same estimate for computing a signature. The prover and verifier time is:

$$
\begin{aligned}
T_1 &\approx 3 \cdot 2^{10} \frac{1}{2^{21}} + 2^{10} \frac{1}{2^{10}} \approx 1 \text{ s}, \\
T_2 &\approx 3 \cdot 2^{15} \frac{1}{2^{21}} + 2^{15} \frac{1}{2^{10}} \approx 32 \text{ s}, \\
T_3 &\approx 3 \cdot 2^{20} \frac{1}{2^{21}} + 2^{20} \frac{1}{2^{10}} \approx 17 \text{ min}.
\end{aligned}
$$

Next, ignoring the signature size, the total proof size is $S = 2(\lambda + 1) \log M$:

$$
\begin{aligned}
S_1 &= 2 \cdot 129 \cdot 10 = 0.32 \text{ KB}, \\
S_2 &= 2 \cdot 129 \cdot 15 = 0.48 \text{ KB}, \\
S_3 &= 2 \cdot 129 \cdot 20 = 0.65 \text{ KB}.
\end{aligned}
$$

Using the signature schemes Dilithium or qTesla both give us subliminal-free signatures a total size ≈ 3 KB, only a factor $1.11 - 1.33$ times larger than the subliminal versions of the signature schemes.

7 Conclusion

We present the first constructions of subliminal-free digital signature schemes that are secure in the post-quantum setting. The core of our construction is based on the combination of verifiable random secrets schemes and the lattice-based signature schemes Dilithium or qTesla.

The lattice-based VRS is of size 13 MB and takes 10 seconds to generate to achieve $\lambda = 128$ bits security. This scheme can easily be made non-interactive if allowing for a l-bit subliminal channel, to the cost of exponential amount of work in l done by the malicious signer. The time it takes to generate the signature is not ideal, but the main bottleneck of the construction is the signature size. It is an open problem to construct a size-efficient post-quantum secure VRS with negligible prover bit-unpredictability.

We relax the soundness of the subliminal-free signature scheme to $1/M$ to achieve a more size-efficient VRS. The signature still provides $\lambda = 128$ bits of security, but the malicious signer has success probability $1/M$ of embedding a subliminal message of the same size as the randomness used in the signature. As a consequence of this relaxation, both the computation performed in the generation and verification of the VRS is linear in M, and hence, M must be small compared to 2^λ. For $M = 2^{10}$ the VRS can produce a signature of size less than 2.5 KB in 1 second, which should be reasonable in most applications that require a subliminal-free channel.

Subliminal digital signatures can be a threat against two-factor authentication systems when the second device is malicious. Boneh et al. [BDC+19] gave a solution to this problem for signatures based on the hardness of computing discrete logarithms over elliptic curves, and our interactive hash-based VRS scheme can be a substitutable replacement for lattice-based signatures.

References

[AAB+19] Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Nina Bindel, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Krämer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. TSubmission to NIST's post-quantum project (2nd round): lattice-based digital signature scheme qTESLA. `https://qtesla.org/wp-content/uploads/2020/04/qTESLA_round2_14.04.2020.pdf`, 2019.

[ABB+20] Erdem Alkim, Paulo S. L. M. Barreto, Nina Bindel, Juliane Kramer, Patrick Longa, and Jefferson E. Ricardini. The Lattice-Based Digital Signature Scheme qTESLA, 2020.

[ABG+] Diego Aranha, Carsten Baum, Kristian Gjøsteen, Tjerand Silde, and Thor Tunge. Electronic Voting using Lattice-Based Commitments and Verifiable Encryption. Unpublished.

[AVPN96] Ross Anderson, Serge Vaudenay, Bart Preneel, and Kaisa Nyberg. The Newton channel. In Ross Anderson, editor, *Information Hiding*, pages 151–156, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, pages 127–144, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[BD91] Mike Burmester and Yvo Desmedt. All Languages in NP Have Divertible Zero-Knowledge Proofs and Arguments Under Cryptographic Assumptions. In Ivan Bjerre Damgård, editor, *Advances in Cryptology — EUROCRYPT '90*, pages 1–10, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[BDC+19] D. Boneh, E. Dauterman, H. Corrigan-Gibbs, D. Mazières, and D. Rizzo. True2F: Backdoor-Resistant Authentication Tokens.

In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 398–416, May 2019.

[BDI+99] Mike Burmester, Yvo G. Desmedt, Toshiya Itoh, Kouichi Sakurai, and Hiroki Shizuya. Divertible and subliminal-free zero-knowledge proofs for languages. *J. Cryptol.*, 12(3):197–223, June 1999.

[BDL+18] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In Dario Catalano and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 368–385, Cham, 2018. Springer International Publishing.

[Beu20] Ward Beullens. Sigma protocols for mq, pkp and sis, and fishy signature schemes. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 183–211, Cham, 2020. Springer International Publishing.

[BGS] Carsten Baum, Kristian Gjøsteen, and Tjerand Silde. Lattice-Based Verifiable Mix-Net. Unpublished.

[BGVS07] Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. A Subliminal-Free Variant of ECDSA. In Jan L. Camenisch, Christian S. Collberg, Neil F. Johnson, and Phil Sallee, editors, *Information Hiding*, pages 375–387, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[Blu83] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *SIGACT News*, 15(1):23–27, January 1983.

[BS05] Jens-Matthias Bohli and Rainer Steinwandt. On subliminal channels in deterministic signature schemes. In Choon-sik Park and Seongtaek Chee, editors, *Information Security and Cryptology – ICISC 2004*, pages 182–194, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[CMY+16] Rongmao Chen, Yi Mu, Guomin Yang, Willy Susilo, Fuchun

Guo, and Mingwu Zhang. Cryptographic Reverse Firewall via Malleable Smooth Projective Hash Functions. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 844–876, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[Des88] Yvo Desmedt. Subliminal-free authentication and signature. In D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, J. Stoer, N. Wirth, and Christoph G. Günther, editors, *Advances in Cryptology — EUROCRYPT '88*, pages 23–33, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[Des96] Yvo Desmedt. Simmons' protocol is not free of subliminal channels. *Proceedings 9th IEEE Computer Security Foundations Workshop*, pages 170–175, 1996.

[DKL+] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTAL-Dilithium. `https://pq-crystals.org/dilithium/data/dilithium-specification-round2.pdf`. Submission to the NIST Post-Quantum Standardization Project, round 2.

[DX10] Qingkuan Dong and Guozhen Xiao. A Subliminal-Free Variant of ECDSA Using Interactive Protocol. In *2010 International Conference on E-Product E-Service and E-Entertainment*, pages 1–3, Nov 2010.

[Eri95] Jeff Erickson. Lower bounds for linear satisfiability problems. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '95, pages 388–395, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

[FS87] Amos Fiat and Adi Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO'*

86, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[Gal20] Herman Galteland. *Malicious Cryptography*. PhD thesis, Norwegian University of Science and Technology, 2020. `https://hdl.handle.net/11250/2649323`.

[GG19] Herman Galteland and Kristian Gjøsteen. Subliminal channels in post-quantum digital signature schemes. Cryptology ePrint Archive, Report 2019/574, 2019. `https://eprint.iacr.org/2019/574`.

[GMR85] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, page 291–304, New York, NY, USA, 1985. Association for Computing Machinery.

[GMR88] Shafi. Goldwasser, Silvio. Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[HAZ17] Alexander Hartl, Robert Annessi, and Tanja Zseby. A Subliminal Channel in EdDSA: Information Leakage with High-Speed Signatures. In *Proceedings of the 2017 International Workshop on Managing Insider Security Threats*, MIST '17, pages 67–78, New York, NY, USA, 2017. ACM.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 525–537, New York, NY, USA, 2018. Association for Computing Machinery.

[LS18] Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications

to lattice-based zero-knowledge proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 204–224, Cham, 2018. Springer International Publishing.

[LWZG10] Dai-Rui Lin, Chih-I Wang, Zhi-Kai Zhang, and D. J. Guan. A digital signature with multiple subliminal channels and its applications. *Comput. Math. Appl.*, 60(2):276–284, July 2010.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, pages 598–616, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 738–755, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[MSD14] Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. Cryptology ePrint Archive, Report 2014/758, 2014. `https://eprint.iacr.org/2014/758`.

[MVR99] Silvio Micali, Salil Vadhan, and Michael Rabin. Verifiable random functions. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, pages 120–, Washington, DC, USA, 1999. IEEE Computer Society.

[Nef01] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, CCS '01, pages 116–125, New York, NY, USA, 2001. ACM.

[NIST17] National Institute Standards and Technology. NIST Post-Quantum Cryptography, Round 1 Submissions. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions`, 2017.

[OO90] Tatsuaki Okamoto and Kazuo Ohta. Divertible zero knowledge interactive proofs and commutative random self-reducibility. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology — EUROCRYPT '89*, pages 134–149, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, 1989.

[Sim84] Gustavus J. Simmons. The prisoners' problem and the subliminal channel. *Advances in Cryptology: Proceedings of Crypto 83*, pages 51–67, 1984.

[Sim85] Gustavus J. Simmons. The subliminal channel and digital signatures. In Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, editors, *Advances in Cryptology*, pages 364–378, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[Sim86] Gustavus J. Simmons. A secure subliminal channel (?). In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 33–41, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[Sim93] G. J. Simmons. An introduction to the mathematics of trust in security protocols. In *[1993] Proceedings Computer Security Foundations Workshop VI*, pages 121–127, June 1993.

[Sim94] Gustavus J. Simmons. Subliminal Communication is Easy Using the DSA. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, pages 218–232, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[Sim98] Gustavus J. Simmons. Results concerning the bandwidth of subliminal channels. *IEEE Journal on Selected Areas in Communications*, 16(4):463–473, May 1998.

[ZL08] Xianfeng Zhao and Ning Li. Reversible watermarking with subliminal channel. In Kaushal Solanki, Kenneth Sullivan, and

Upamanyu Madhow, editors, *Information Hiding*, pages 118–131, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[ZLLZ13] Yinghui Zhang, Hui Li, Xiaoqing Li, and Hui Zhu. Provably Secure and Subliminal-Free Variant of Schnorr Signature. In Khabib Mustofa, Erich J. Neuhold, A. Min Tjoa, Edgar Weippl, and Ilsun You, editors, *Information and Communication Technology*, pages 383–391, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.