Verifiable Random Secrets and Subliminal-Free Digital Signatures

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 the first constructions in the post-quantum setting.
Our first construction is a lattice-based scheme using the commitment scheme from Baum et al. (SCN 2018) and a zero-knowledge proof of shuffle of known content to generate verifiable randomness. This can be combined with the signature framework by Lyubashevsky (EUROCRYPT 2012) to achieve subliminal-free signatures. The concrete instantiation takes 5 seconds to create a subliminal-free signature of size 12.65 MB.
Our second 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 Schnorr-like signature scheme. Our most practical instantiation only takes 1 second to generate a subliminal-free lattice-based signature with overhead 0.6 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. (S&P 2019) gave a solution to this problem based on discrete logarithms. Our protocol can be a drop-in solution for lattice-based signatures.

Keywords: post-quantum cryptography · subliminal channels · digital signatures · verifiable random secrets · zero-knowledge proofs

1 Introduction

A *subliminal channel* is a covert communication channel inserted into an overt channel. Over the subliminal channel two or more principals can send *subliminal messages* to each other that are oblivious to anyone else that do not have knowledge of this channel. Simmons introduced subliminal channels as a solution to his Prisoners' Problem [33], where two prisoners wants to communicate covertly over a channel controlled by a warden. Using a subliminal channel the prisoners encode and decode messages into the overt channel, possibly using pre-shared information, in such a way that the warden is oblivious of the subliminal channel.

The existence of subliminal channels pose a threat to anyone that wants to prevent sensitive information being stolen and reach malicious parties, such as companies or governmental entities. For example, a bank keeps credit card credentials stored on a secure storage device, a company stores their intellectual properties on a company server, or a hospital keeps medical data and personal information about patients in electronic journals. In all of these examples a malicious insider can use a subliminal channel to secretly leak sensitive information, or keys to access this information, without a network administrator, monitoring access to this information, noticing the theft.

A popular security measure to prevent phishing and software compromise is two-factor authentication. This can be done via an app on the phone or a hardware token being connected to the computer. In the case of a malicious device, it can use a subliminal channel in the signature to leak information about your key, so that a third party can impersonate you later. Boneh et al. [9] give a construction called **True2F** that creates a subliminal-free channel for ECDSA.

In this paper we will focus creating a subliminal-free signature scheme. For classical adversaries, constructing these types of schemes is solved [7–9, 16, 39]. We focus on constructing subliminal-free digital signature schemes that are secure against a quantum adversary.

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`, where `S` and `R` sends signed plaintext messages to each other and `W` will prevent subliminal messages.

Before a signature is generated, `W` and `S` interact to jointly produce the random value used in the signature. The sender must then provide a proof showing the random value was used in the signature. The sender 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 sender will be able to choose his own public and private signing keys, however, `S` will not be able to update any keys during the signing process. The warden will abort if any signature is invalid, with respect to the public key of `S`, and close the channel. If `S` aborts during the signing process, e.g. if the signature does not include the subliminal bits and he wants to re-try, `W` 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, or the signing key. The symmetric key can be used to encrypt the subliminal messages to make them indistinguishable from random values, and the signing key can be used to recover subliminal messages.

We are not interested in hiding information in the messages themselves, called 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. [7], 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 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. [28]. The main difference between a VRF and a VRS is that the random number produced in a VRS is secret to the verifier.

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 [19], 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 [25, 26] 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 [33]. Simmons showed that subliminal channels in digital signature schemes exist [34–36] and since more have been found [3, 8, 20, 22, 24, 40].

As a response to Simmons's new notion, subliminal-free digital signatures schemes was made [7, 8, 14, 16, 32, 36, 37, 39]. Bohli et al. introduced *subliminal-free with proof signature schemes*, where the sender of a signature sends a proof to the warden, and only to the warden, which proves the signature sent is subliminal-free [7]. In *divertible protocols* [6, 10, 11, 31] a third party can be inserted between the two communicators, where the third party can remove or detect subliminal messages. A *cryptographic reverse firewall* [12, 29] 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 [38], and none of the posposed schemes of the second round are subliminal free [20]. The CRYSTAL-Dilithium [17], qTesla [2] and other submissions could potentially become subliminal-free using our techniques.

2 Preliminaries

2.1 Polynomial Rings and Norms

Let $N \geq \delta > 1$ be powers of 2 and p a prime congruent to $2\delta + 1 \mod 4\delta$. Then we define the rings $R = \mathbb{Z}[X]/\langle X^N + 1\rangle$ and $R_p = R/pR$, 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.

2.2 Discrete Gaussian Distribution

The continuous normal distribution over $\mathbb{R}^k$, k-dimensional vectors of the real numbers centered at $\boldsymbol{v} \in \mathbb{R}^k$ with standard deviation σ, is given by

$$\rho(\boldsymbol{x})^k_{\boldsymbol{v},\sigma} = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-||\boldsymbol{x} - \boldsymbol{v}||_2^2}{2\sigma^2}\right).$$

The *discrete Gaussian distribution* over the ring R is achieved by normalizing the continuous distribution over R, for $\boldsymbol{x} \in R$ represented as a N dimensional vector, by letting

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

where $\rho^N_\sigma(R) = \sum_{\boldsymbol{x} \in R} \rho^N_\sigma(\boldsymbol{x})$. 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. However, rejection sampling can still be very expensive, and it's a randomized procedure. If we do some pre-computations, we can use the algorithm given in the qTelsa-scheme [2] to generate the discrete Gaussians deterministically from a uniformly random seed. This is specified in Algorithm 11 in the NIST-submission of qTelsa [1].

2.3 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. The fastest algorithms for solving k-SUM runs in $\mathcal{O}(n^{k/2})$ [18].

3 Lattice-Based Commitments and Zero-Knowledge Proofs

Lattice-Based Commitments. The commitment scheme by Baum et al. [4] 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

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

- `Com`, on input a message $m \in R_p$, samples an $\boldsymbol{r} \xleftarrow{\$} R_p^3$ where $\|\boldsymbol{r}\|_\infty = 1$, 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 $\boldsymbol{c}$ together with $\boldsymbol{d} = (m; \boldsymbol{r}, 1)$,
- `Open`, on input $\boldsymbol{c}$ and $(m, \boldsymbol{r}, f)$, with $\|f\|_\infty \leq 2$ and $\|f\|_2 \leq 72$, 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 $\sigma = 11 \cdot 36 \cdot \sqrt{3N}$. It outputs 1 if all these conditions hold, and 0 otherwise.

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

$$[x_1] = \mathtt{Com}(x_1; \boldsymbol{r}) = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, [x_2] = \mathtt{Com}(x_2; \boldsymbol{r}') = \begin{bmatrix} c_1' \\ c_2' \end{bmatrix}, [x_3] = \mathtt{Com}(x_3; \boldsymbol{r}'') = \begin{bmatrix} c_1'' \\ c_2'' \end{bmatrix},$$

where $x_3 = \alpha_1 x_1 + \alpha_2 x_2$ for some public values $\alpha_1, \alpha_2 \in R_p$. Then Baum et al. [4] gives a zero-knowledge proof of knowledge protocol Π_{Lin} of such relations. This protocol can easily be extended to prove linear relations between an arbitrary number of commitments. Let $\Pi_{\text{Lin}}(([x_1], [x_2]), [x_3], (\alpha_1, \alpha_2))$ denote a run of the protocol producing a proof π_L, and let $\Pi_{\text{LinV}}(([x_1], [x_2]), [x_3], (\alpha_1, \alpha_2), \pi_L)$ denote the verification of this proof. If $\alpha_1 = \alpha_2 = 1$, then the scalars are omitted. The detailed protocol can be found in Appendix A.

Zero-Knowledge Proof of Correct Shuffle. In an unpublished work we give an efficient protocol Π_{Shuffle} for a Neff-like [30] shuffle of known values for the lattice-based commitments by Baum et al. [4]. Given a list of elements $(\hat{M}_1, \hat{M}_2, \dots, \hat{M}_\tau)$ from R_p and commitments $([M]_1, [M]_2, \dots, [M]_\tau)$, 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 $\Pi_{\text{Shuffle}}(\{[M]_i\}, \{\hat{M}_i\}, \gamma)$ denote a run of the shuffle-protocol producing a proof π_S, and let $\Pi_{\text{ShuffleV}}(\{[M]_i\}, \{\hat{M}_i\}, \pi_S)$ denote the verification of this proof.

For completeness, we present the full shuffle-protocol in Appendix B and estimate the size and timings of the protocol in Appendix C. We could use the shuffle by Costa et al. [13] instead, but their protocol is less inefficient compared to ours, and they do not provide an estimate for size and timings.

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 desirable 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.

Definition 1 (Verifiable Random Secrets). *A Verifiable Random Secret-scheme (VRS) consists of a function $r(\cdot,\cdot)$, an interactive protocol seed ($\mathit{\Pi}_{\texttt{Seed}}$) 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`*,*
- $\mathit{\Pi}_{\texttt{Seed}}$*, 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 `sp` *is an implicit input to all algorithms following* `Setup`*.*

The interactive VRS is visualized in Figure 1. The first thing we require from a VRS is that it is *complete*. Further, we use games to define the security of the scheme, and continue by defining *Binding*, *Prover bit-Unpredictability*, *Honest-Verifier Secrecy* and *ϵ-Security.*

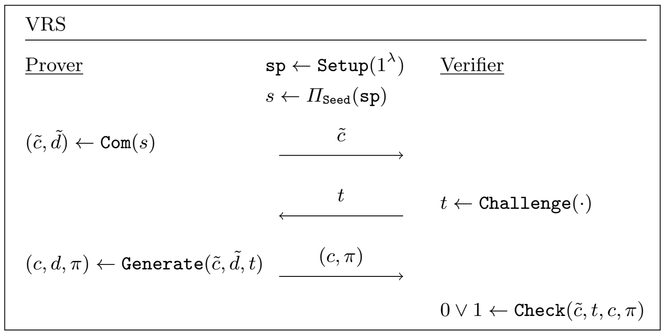
VRS
Prover
$\texttt{sp} \leftarrow \texttt{Setup}(1^\lambda)$
Verifier
$s \leftarrow \mathit{\Pi}_{\texttt{Seed}}(\texttt{sp})$
$(\tilde{c}, \tilde{d}) \leftarrow \texttt{Com}(s)$
$\tilde{c}$
t
$t \leftarrow \texttt{Challenge}(\cdot)$
$(c, d, \pi) \leftarrow \texttt{Generate}(\tilde{c}, \tilde{d}, t)$
(c, π)
$0 \vee 1 \leftarrow \texttt{Check}(\tilde{c}, t, c, \pi)$

Fig. 1: Our abstract verifiable random secret scheme.

Definition 2 (Binding). *We say that the VRS is* binding*, if a cheating prover* $\texttt{P}^*$ *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 run of the* *protocol, depending on the commitment $\tilde{c}$ and the challenge t. We define the binding advantage $\mathbf{Adv^B}$ of the cheating prover $\mathtt{P}^*$ to be:*

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

Definition 3 (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 $\mathbf{Adv_f^{PbU}}$ of a cheating prover $\mathtt{P}^*$, with respect to f, to be:*

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

Definition 4 (Honest-Verifier Secrecy). *We say that the VRS has* honest-verifier secrecy, *if a honest but curious verifier $\mathtt{V}^*$ is unable to distinguish between a honestly generated value r and a random r sampled from the set $\mathtt{R} = \{r(s,t) : s \leftarrow \mathit{\Pi}_{\mathtt{Seed}}, t \leftarrow \mathtt{Challenge}\}$, where r is a function of the seed s and the challenge t. We define the honest-verifier secrecy advantage $\mathbf{Adv^{HVS}}$ of a verifier $\mathtt{V}^*$ as:*

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

Definition 5 (ϵ-Security). *A VRS is said to be ϵ-secure, for an ϵ negligible in the security parameter λ, if each of the advantages are less than ϵ. That is, we have ϵ-security if*

$$\mathbf{Adv^B}(\mathtt{P}^*) \leq \epsilon(\lambda), \qquad \mathbf{Adv_f^{PbU}}(\mathtt{P}^*) \leq \epsilon(\lambda), \qquad \mathbf{Adv_f^{HVS}}(\mathtt{V}^*) \leq \epsilon(\lambda).$$

5 Subliminal-Free Digital Signatures

5.1 How to Achieve a Subliminal-Free Channel?

Let $\mathtt{S}$ be the sender, $\mathtt{R}$ be the receiver and $\mathtt{W}$ the warden. We consider $\mathtt{S}$, $\mathtt{R}$ and $\mathtt{W}$ 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 – 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. 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, we must restrict S's control over the choice of randomness used in the signatures.

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. If S gets to decide if they want to send the message or not 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 [15]. It follows that the warden cannot allow S to abort if he wants the channel to be subliminal-free.

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 6 (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 $\mathcal{S}$ 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 W. That is,*

- $\mathtt{Encode}_{\mathtt{S}^*}$ *on input the message m, signing key* 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 7 (Subliminal-Free Digital Signatures). *Let m be a fixed message, $\mathcal{S}$ a signature scheme, $\hat{m}$ a subliminal message chosen by* S *and s some secret pre-shared information between* S *and* 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* S *and* W*. Assume that* W *will close the channel if* S *deviates from the protocol. Then we say that $\mathcal{S}$ is a subliminal-free digital signature scheme if* S *cannot reliably both create a proof π accepted by* W *and at the same time encode a subliminal message $\hat{m}$ into σ that can be decoded by* R *but cannot be decoded nor detected by* W*.*

However, if $\mathcal{S}$ is a probabilistic signature scheme used in the non-interactive setting, then 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. See definition in Appendix D.

5.3 Subliminal-Free Digital Signature Scheme

The following definition is similar but not equal to the subliminal-free digital signatures with proof definition by Bohli et al. [7].

Definition 8 (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.*

5.4 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*, but we skip the definition of completeness and move the (extended) definition of *existential forgery* to Appendix E.

Definition 9 (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* $\texttt{S}^*$*, 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* $\texttt{R}^*$ *can decode. We also require* $\texttt{S}^*$ *to produce a proof* π *of correctness accepted by the warden* `W`*. That is, the subliminal-free signature scheme is sound if*

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

where the algorithms $\texttt{Encode}_{\texttt{S}^*}$ *and* $\texttt{Decode}_{\texttt{R}^*}$ *are as in Definition 6.*

6 Our Schemes

6.1 Lattice-Based Subliminal-Free Digital Signatures

Our lattice-based VRS scheme is built on top of the commitment scheme [4] combined with the verifiable shuffle of known content from Section 3.

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.

Theorem 1. *The lattice-based VRS is complete and ϵ-Secure.*

Proof. We argue why the protocol is secure, and will include a formal proof in the full version. Completeness is trivial and binding follows from the binding property of the commitment scheme. Prover bit-unpredictability follows from the fact that each r_j is a random sum of κ random elements, and is unpredictable for the prover until the indices are given by the verifier. Honest-verifier secrecy follows from k-SUM, given that the commitment scheme is hiding: the verifier is given a commitment to the sum of κ out of τ possible elements. Even if the value is given in the clear, the verifier is not able to decide if this is the underlying message of the commitment (breaking hiding) or to determine if this value even is a sum of κ out of the τ values initially given to the prover (deciding k-SUM).

The VRS scheme is visualized as a part of the subliminal-free digital signature scheme in Figure 2. The lattice-based VRS give us three commitments c_1, c_2, c_3 that are known to the warden `W`, and valid openings $(y_1, \boldsymbol{r}_1), (y_2, \boldsymbol{r}_2), (y_3, \boldsymbol{r}_3)$ known to the sender `S`. All the y_i's are Gaussian distributed elements from R_p with standard deviation σ. Finally, `S` computes the signature, attach a proof of linearity for this statement, and send the message m and the signature $(t', \boldsymbol{z})$ together with the output of the VRS to `W`, which then verifies that the proofs and signature are correct. If, and only if, both the `Check`- and `Verify` algorithms outputs 1, `W` forwards the message and signature to the receiver `R`.

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

Proof. The soundness of the protocol follows from the prover bit-unpredictability of the VRS, and the security against existential forgeability follows from the honest-verifier secrecy of the VRS combined with the security against existential forgeability of the signature scheme. See full version for a formal security proof.

Lattice-Based Subliminal-Free Signature Scheme

Prover		Verifier
		Seed:
	$\xleftarrow{s = \{s_i\}}$	$s_i \overset{\$}{\leftarrow} \mathcal{N}_{\sigma/\sqrt{\kappa}}, 1 \leq i \leq \tau$
Com:		
$\gamma \overset{\$}{\leftarrow} S_\tau$		
$(\tilde{c}_i, \tilde{d}_i) \leftarrow \texttt{Com}(s_{\gamma(i)})$	$\xrightarrow{\tilde{c} = \{\tilde{c}_i\}}$	
		Challenge:
$\pi_S \leftarrow \mathit{\Pi}_{\text{Shuffle}}(\{\tilde{c}_i\}, \{s_i\}, \gamma)$		$T_j \overset{\$}{\subset} \{1, \ldots, \tau\}$,
	$\xleftarrow{t = \{T_j\}}$	$\|T_j\| = \kappa, 1 \leq j \leq 3$
Generate:		
$(c_j, d_j) \leftarrow \sum_{l \in T_j} \texttt{Com}(s_{\gamma^{-1}(l)})$		
$(t', \boldsymbol{z}) \leftarrow \texttt{Sign}(m, \texttt{sk})$		
$\pi_L \leftarrow \mathit{\Pi}_{\text{Lin}}(\{c_j\}, t')$	$\xrightarrow{(m, (t', \boldsymbol{z})), (\{c_j\}, (\pi_S, \pi_L))}$	
		Check:
		$1 \overset{?}{=} \mathit{\Pi}_{\text{ShuffleV}}(\{\tilde{c}_i\}, \{s_i\}, \pi_S)$
		$1 \overset{?}{=} \mathit{\Pi}_{\text{LinV}}(\{c_j\}, t', \pi_L)$
		Verify:
		$1 \overset{?}{=} \texttt{Verify}(\texttt{vk}, m, (t', \boldsymbol{z}))$
		If all algorithms output 1:
		Send $(m, (t', \boldsymbol{z}))$ to the receiver.

Fig. 2: The lattice-based subliminal-free digital signature scheme, using the lattice-based VRS and a lattice signature scheme based on Lyubashevsky [26].

6.2 Subliminal-Free Digital Signatures from One-Way Functions

Let $\mathtt{H}$ be a cryptographic hash-function with output length 3λ, and let $\mathtt{OWF}$ be a one-way function used in any Schnorr-like 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 by Katz et al. [23] and Beullens [5].

Let s be a binary seed of length 3λ, 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 $h_1 = \mathtt{H}(s, 2)$, and so on. 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 3λ.

In the case that the subsequent application, e.g. a subliminal-free Schnorr-like 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 a deterministic algorithm that produces the correct distribution. To produce similar lattice-based signatures as in the previous construction, we can evaluate the algorithm $\mathtt{GaussSampler}_\sigma$ on the leaf nodes to produce the desired outcome. Let's denote the new values by $v_0, ..., v_{M-1}$.

Then we apply the one-way function $\mathtt{OWF}$ on the values $v_0, ..., v_{M-1}$ to produce $w_0, ..., w_{M-1}$, where $w_i = \mathtt{OWF}(v_i)$ for $0 \leq i \leq M-1$. For 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 for the signature scheme.

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 $c = w_I = \mathtt{OWF}(v_I)$ 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.

The full VRS protocol is illustrated as a part of Figure 3 and works as follows:

1. $\mathtt{Seed}$: $\mathtt{P}$ chose a random bit string s of length 3λ and keeps this private.
2. $\mathtt{Commit}$: $\mathtt{P}$ generates the full tree applying the algorithm $\mathtt{BuildTree}$ on s, performing the procedure described above, and sends the root $\tilde{c}$ to $\mathtt{V}$.
3. $\mathtt{Challenge}$: $\mathtt{V}$ draws a random index $t = I$, $0 \leq I \leq M-1$, and sends t to $\mathtt{P}$.
4. $\mathtt{Generate}$: $\mathtt{P}$ publishes $c = w_I$ and the proof π_I, generated by applying the algorithm $\mathtt{SubTrees}$ on s and I, which computes the roots of the 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, which re-computes the full tree based on the given information, and compares the root to $\tilde{c}$.

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

Proof. Completeness follows directly from the construction of the Merkle-tree. Binding follows from the binding property of the hash-function, and honest-verifier secrecy follows from the unpredictable output of the hash-function, modeled as a random oracle. Further, prover bit-unpredictability also follows from the unpredictable output of the hash-function combined with the size of the Merkle-tree. If the prover is able to guess the index of the chosen node in advance, he can replace that node with any value of his choice while constructing the tree, and he'll win the game with probability one. Hence, the chance of winning the game is $1/M$, where M is the number of leaf nodes in the tree. Setting M to be exponential in λ, we get that the VRS is ϵ-secure.

Let $\mathcal{S}$ be a Schnorr-like-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.

Theorem 4. *The SFS scheme in Figure 3 is complete, sound and secure against existential forgability.*

Proof. The soundness of the protocol follows from the prover bit-unpredictability of the VRS, and the security against existential forgeability follows from the honest-verifier secrecy of the VRS combined with the security against existential forgeability of the signature scheme. See full version for a formal security proof. Note that soundness does not affect the security against existential forgability attacks, and hence, we can offer tradeoffs between soundness, timing and size.

6.3 Efficiency and Size

We present the size and timings of the two schemes in Table 1. The cheating probability for including subliminal information may vary, but the long term security of all schemes are $\lambda = 128$ bits. If the Lattice-SFS scheme is interactive, with challenges ensuring 128 bit security, we get a subliminal-free scheme with cheating probability 2^{-128}. If the scheme is non-interactive, a cheating prover may create a l-subliminal channel if he run the protocol locally approximately 2^l times to ensure that the last l bits of the randomness is of his choice. In our second construction, we note that the running time of the hash-based VRS is dependent on the number of leaf nodes in the tree, which is inversely proportional with the soundness of the signature scheme. However, when running the protocol interactively, soundness of 2^{-10} or 2^{-20} is good enough for most applications, and hence, we can get a very practical scheme, both in terms of size and speed, by allowing for larger soundness. More details can be found in Appendix F. Both Dilithium or qTesla offers signatures of size $\approx 3.3 - 3.7$ KB.

Hash-Based Subliminal-Free Signature Scheme		
Prover		Verifier
`Seed`:		
$s \xleftarrow{\$} \{0,1\}^{3\lambda}$		
`Com`:		
$(\tilde{c}, \tilde{d}) \leftarrow \texttt{BuildTree}(s)$	$\xrightarrow{\tilde{c}}$	`Challenge`:
`Generate`:	$\xleftarrow{t = I}$	$I \xleftarrow{\$} \{0, ..., M-1\}$
$(c, d) \leftarrow (w_I, v_I)$		
$\pi_I \leftarrow \texttt{SubTrees}(s, I)$		
$(w_I, z) \leftarrow \texttt{Sign}(m, \texttt{sk})$	$\xrightarrow{(c, \pi_I, (w_I, z))}$	`Check`:
		$\tilde{c} \overset{?}{=} \texttt{CompleteTree}(w_I, \pi_I)$
		`Verify`:
		$1 \overset{?}{=} \texttt{Verify}(\texttt{vk}, m, (w_I, z)))$
		If all algorithms output 1:
		Send $(m, (w_I, z))$ to the receiver.

Fig. 3: The hash-based subliminal-free digital signature scheme, using the hash-based VRS and a Schnorr-like signature scheme.

Finally, we note that the randomness created in the VRS, might not pass the rejection sampling in the signature scheme. For Dilithium and qTesla we expect up to three trials to create a signature, and hence, we may allow a sender to run the VRS up to three times with abort before sending a signature. This allows for a subliminal 1-bit channel with 87.5% likelihood of success, which would be considered somewhat reliable, e.g by using a error-correcting code. If the signer abort too many times, warden closes the channel completely.

Scheme	Overhead	Time	l-free	Cheating probability
Lattice-SFS	12.65 MB	≈ 5 s	Yes / No	2^{-128}
Hash-SFS-10	0.6 KB	≈ 1 s	1 bit	2^{-10}
Hash-SFS-15	0.8 KB	≈ 33 s	1 bit	2^{-15}
Hash-SFS-20	1 KB	≈ 18 min	1 bit	2^{-20}

Table 1: Size, efficiency and properties of the subliminal-free signature schemes.

Bibliography

[1] Akleylek, S., Alkim, E., Barreto, P.S.L.M., Bindel, N., Buchmann, J., Eaton, E., Gutoski, G., Krämer, J., Longa, P., Polat, H., Ricardini, J.E., Zanon, G.: 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)

[2] Alkim, E., Barreto, P.S.L.M., Bindel, N., Kramer, J., Longa, P., Ricardini, J.E.: The Lattice-Based Digital Signature Scheme qTESLA (2020)

[3] Anderson, R., Vaudenay, S., Preneel, B., Nyberg, K.: The Newton channel. In: Anderson, R. (ed.) Information Hiding. pp. 151–156. Springer Berlin Heidelberg, Berlin, Heidelberg (1996)

[4] Baum, C., Damgård, I., Lyubashevsky, V., Oechsner, S., Peikert, C.: More efficient commitments from structured lattice assumptions. In: Catalano, D., De Prisco, R. (eds.) Security and Cryptography for Networks. pp. 368–385. Springer International Publishing, Cham (2018)

[5] Beullens, W.: Sigma protocols for mq, pkp and sis, and fishy signature schemes. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020. pp. 183–211. Springer International Publishing, Cham (2020)

[6] Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) Advances in Cryptology — EUROCRYPT'98. pp. 127–144. Springer Berlin Heidelberg, Berlin, Heidelberg (1998)

[7] Bohli, J.M., González Vasco, M.I., Steinwandt, R.: A Subliminal-Free Variant of ECDSA. In: Camenisch, J.L., Collberg, C.S., Johnson, N.F., Sallee, P. (eds.) Information Hiding. pp. 375–387. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[8] Bohli, J.M., Steinwandt, R.: On subliminal channels in deterministic signature schemes. In: Park, C.s., Chee, S. (eds.) Information Security and Cryptology – ICISC 2004. pp. 182–194. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)

[9] Boneh, D., Dauterman, E., Corrigan-Gibbs, H., Mazières, D., Rizzo, D.: True2F: Backdoor-Resistant Authentication Tokens. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 398–416 (May 2019). https://doi.org/10.1109/SP.2019.00048

[10] Burmester, M., Desmedt, Y.: All Languages in NP Have Divertible Zero-Knowledge Proofs and Arguments Under Cryptographic Assumptions. In: Damgård, I.B. (ed.) Advances in Cryptology — EUROCRYPT '90. pp. 1–10. Springer Berlin Heidelberg, Berlin, Heidelberg (1991)

[11] Burmester, M., Desmedt, Y.G., Itoh, T., Sakurai, K., Shizuya, H.: Divertible and subliminal-free zero-knowledge proofs for languages. J. Cryptol. **12**(3), 197–223 (Jun 1999), `http://dx.doi.org/10.1007/s001459900053`

[12] Chen, R., Mu, Y., Yang, G., Susilo, W., Guo, F., Zhang, M.: Cryptographic Reverse Firewall via Malleable Smooth Projective Hash Functions. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016. pp. 844–876. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)

[13] Costa, N., Martínez, R., Morillo, P.: Lattice-based proof of a shuffle. In: Bracciali, A., Clark, J., Pintore, F., Rønne, P.B., Sala, M. (eds.) Financial Cryptography and Data Security. pp. 330–346. Springer International Publishing, Cham (2020)

[14] Desmedt, Y.: Subliminal-free authentication and signature. In: Barstow, D., Brauer, W., Brinch Hansen, P., Gries, D., Luckham, D., Moler, C., Pnueli, A., Seegmüller, G., Stoer, J., Wirth, N., Günther, C.G. (eds.) Advances in Cryptology — EUROCRYPT '88. pp. 23–33. Springer Berlin Heidelberg, Berlin, Heidelberg (1988)

[15] Desmedt, Y.: Simmons' protocol is not free of subliminal channels. Proceedings 9th IEEE Computer Security Foundations Workshop pp. 170–175 (1996)

[16] Dong, Q., Xiao, G.: A Subliminal-Free Variant of ECDSA Using Interactive Protocol. In: 2010 International Conference on E-Product E-Service and E-Entertainment. pp. 1–3 (Nov 2010). https://doi.org/10.1109/ICEEE.2010.5660874

[17] Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTAL-Dilithium. `https://pq-crystals.org/dilithium/data/dilithium-specification-round2.pdf`, Submission to the NIST Post-Quantum Standardization Project, round 2

[18] Erickson, J.: Lower bounds for linear satisfiability problems. In: Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 388–395. SODA '95, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1995)

[19] Fiat, A., Shamir, A.: How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology — CRYPTO' 86. pp. 186–194. Springer Berlin Heidelberg, Berlin, Heidelberg (1987)

[20] Galteland, H., Gjøsteen, K.: Subliminal channels in post-quantum digital signature schemes. Cryptology ePrint Archive, Report 2019/574 (2019), `https://eprint.iacr.org/2019/574`

[21] Hart, W., Johansson, F., Pancratz, S.: FLINT: Fast Library for Number Theory (2013), version 2.4.0, `http://flintlib.org`

[22] Hartl, A., Annessi, R., Zseby, T.: A Subliminal Channel in EdDSA: Information Leakage with High-Speed Signatures. In: Proceedings of the 2017 International Workshop on Managing Insider Security Threats. pp. 67–78. MIST '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3139923.3139925, `http://doi.acm.org/10.1145/3139923.3139925`

[23] Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Se-

curity. p. 525–537. CCS '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3243734.3243805, `https://doi.org/10.1145/3243734.3243805`

[24] Lin, D.R., Wang, C.I., Zhang, Z.K., Guan, D.J.: A digital signature with multiple subliminal channels and its applications. Comput. Math. Appl. **60**(2), 276–284 (Jul 2010). https://doi.org/10.1016/j.camwa.2010.01.001

[25] Lyubashevsky, V.: Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In: Matsui, M. (ed.) Advances in Cryptology – ASIACRYPT 2009. pp. 598–616. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)

[26] Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. pp. 738–755. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

[27] Lyubashevsky, V., Seiler, G.: Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018. pp. 204–224. Springer International Publishing, Cham (2018)

[28] Micali, S., Vadhan, S., Rabin, M.: Verifiable random functions. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science. pp. 120–. FOCS '99, IEEE Computer Society, Washington, DC, USA (1999)

[29] Mironov, I., Stephens-Davidowitz, N.: Cryptographic reverse firewalls. Cryptology ePrint Archive, Report 2014/758 (2014), `https://eprint.iacr.org/2014/758`

[30] Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: Proceedings of the 8th ACM Conference on Computer and Communications Security. pp. 116–125. CCS '01, ACM, New York, NY, USA (2001). https://doi.org/10.1145/501983.502000, `http://doi.acm.org/10.1145/501983.502000`

[31] Okamoto, T., Ohta, K.: Divertible zero knowledge interactive proofs and commutative random self-reducibility. In: Quisquater, J.J., Vandewalle, J. (eds.) Advances in Cryptology — EUROCRYPT '89. pp. 134–149. Springer Berlin Heidelberg, Berlin, Heidelberg (1990)

[32] Simmons, G.J.: An introduction to the mathematics of trust in security protocols. In: [1993] Proceedings Computer Security Foundations Workshop VI. pp. 121–127 (June 1993). https://doi.org/10.1109/CSFW.1993.246634

[33] Simmons, G.J.: The prisoners' problem and the subliminal channel. Advances in Cryptology: Proceedings of Crypto 83 pp. 51–67 (1984). https://doi.org/10.1007/978-1-4684-4730-9_5

[34] Simmons, G.J.: The subliminal channel and digital signatures. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) Advances in Cryptology. pp. 364–378. Springer Berlin Heidelberg, Berlin, Heidelberg (1985)

[35] Simmons, G.J.: A secure subliminal channel (?). In: Williams, H.C. (ed.) Advances in Cryptology — CRYPTO '85 Proceedings. pp. 33–41. Springer Berlin Heidelberg, Berlin, Heidelberg (1986)

[36] Simmons, G.J.: Subliminal Communication is Easy Using the DSA. In: Helleseth, T. (ed.) Advances in Cryptology — EUROCRYPT '93. pp. 218–232. Springer Berlin Heidelberg, Berlin, Heidelberg (1994)

[37] Simmons, G.J.: Results concerning the bandwidth of subliminal channels. IEEE Journal on Selected Areas in Communications **16**(4), 463–473 (May 1998). https://doi.org/10.1109/49.668970

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

[39] Zhang, Y., Li, H., Li, X., Zhu, H.: Provably Secure and Subliminal-Free Variant of Schnorr Signature. In: Mustofa, K., Neuhold, E.J., Tjoa, A.M., Weippl, E., You, I. (eds.) Information and Communication Technology. pp. 383–391. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)

[40] Zhao, X., Li, N.: Reversible watermarking with subliminal channel. In: Solanki, K., Sullivan, K., Madhow, U. (eds.) Information Hiding. pp. 118–131. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

A Zero-Knowledge Proof of Linear Relations

The Π_{Lin} in Figure 4 is a zero-knowledge proof of knowledge of the linear relation $x_3 = \alpha_1 x_1 + \alpha_2 x_2$. In fact it is an adapted version of the linearity proof in [4]. The verification protocol is given in Figure 5.

Π_{Lin}

$$\begin{array}{lcr}
\underline{\text{Prover}} & & \underline{\text{Verifier}} \\
\boldsymbol{y}, \boldsymbol{y}', \boldsymbol{y}'' \leftarrow \mathcal{N}_\sigma^3 & & \\
t \leftarrow \langle \boldsymbol{a}, \boldsymbol{y} \rangle & & \\
t' \leftarrow \langle \boldsymbol{a}, \boldsymbol{y}' \rangle & & \\
t'' \leftarrow \langle \boldsymbol{a}, \boldsymbol{y}'' \rangle & & \\
u \leftarrow \alpha_1 \langle \boldsymbol{a}', \boldsymbol{y} \rangle + \alpha_2 \langle \boldsymbol{a}', \boldsymbol{y}' \rangle - \langle \boldsymbol{a}', \boldsymbol{y}'' \rangle & \xrightarrow{t, t', t'', u} & \\
 & \xleftarrow{\beta} & \beta \leftarrow \mathcal{C} \\
\boldsymbol{z} \leftarrow \boldsymbol{y} + \beta \boldsymbol{r} & & \\
\boldsymbol{z}' \leftarrow \boldsymbol{y}' + \beta \boldsymbol{r}' & & \\
\boldsymbol{z}'' \leftarrow \boldsymbol{y}'' + \beta \boldsymbol{r}'' & & \\
\text{Abort with probability:} & & \\
1 - \min\left(1, \dfrac{\mathcal{N}_\sigma^9(\boldsymbol{z}||\boldsymbol{z}'||\boldsymbol{z}'')}{\frac{1}{3} \cdot \mathcal{N}^9_{\beta(\boldsymbol{r}||\boldsymbol{r}'||\boldsymbol{r}''),\sigma}(\boldsymbol{z}||\boldsymbol{z}'||\boldsymbol{z}'')}\right) & \xrightarrow{\boldsymbol{z}, \boldsymbol{z}', \boldsymbol{z}''} & \\
 & & \Pi_{\text{LinV}}
\end{array}$$

Fig. 4: Protocol Π_{Lin} is a sigma-protocol to prove the relation $x_3 = \alpha_1 x_1 + \alpha_2 x_2$, given the commitments $[x] = \texttt{Com}(x; \boldsymbol{r})$, $[x'] = \texttt{Com}(x'; \boldsymbol{r}')$ and the scalars α_1, α_2.

B Zero-Knowledge Proof of Correct Shuffle

In this section we present the shuffle protocol. We construct a public-coin $4 + 3\tau$-move protocol* such that the commit-challenge-response stages require the prover to solve a system of linear equations in order to prove a correct shuffle. Our construction extends Neff's construction [30] to the realm of post-quantum assumptions.

The proof of shuffle protocol will use the commitments defined in Section 3. For the shuffle proof to work, the prover $\mathcal{P}$ and verifier $\mathcal{V}$ receive commit-

*This is only a theoretical problem as the protocol is public-coin and can therefore directly be transformed into NIZKs using the Fiat-Shamir transform.

```
$\Pi_{\mathrm{LinV}}$
Verifier
return Accept iff
1 : $\|\boldsymbol{z}_i\|, \|\boldsymbol{z}_i'\|, \|\boldsymbol{z}_i''\| \overset{?}{\leq} 2\sigma\sqrt{N}$
2 : $\langle \boldsymbol{a}, \boldsymbol{z} \rangle \overset{?}{=} t + \beta c_1$
3 : $\langle \boldsymbol{a}, \boldsymbol{z}' \rangle \overset{?}{=} t' + \beta c_1'$
4 : $\langle \boldsymbol{a}, \boldsymbol{z}'' \rangle \overset{?}{=} t'' + \beta c_1''$
5 : $\alpha_1 \langle \boldsymbol{a}', \boldsymbol{z} \rangle + \alpha_2 \langle \boldsymbol{a}', \boldsymbol{z}' \rangle - \langle \boldsymbol{a}', \boldsymbol{z}'' \rangle \overset{?}{=} (\alpha_1 c_2 + \alpha_2 c_2' - c_2'')\beta + u$
```

Fig. 5: Verification protocol Π_{LinV} for the Π_{Lin}-protocol.

ments $\{[m_i]\}_{i=1}^{\tau}$. $\mathcal{P}$ also receives the set of openings $\{(m_i, \boldsymbol{r}_i)\}_{i=1}^{\tau}$ as well as a permutation $\gamma \in S_\tau$, while $\mathcal{V}$ obtains messages $\{\hat{m}_i\}_{i=1}^{\tau}$ where $\hat{m}_i = m_{\gamma^{-1}(i)}$.

The goal is to ensure that the following relation R_{Shuffle} holds:

$$R_{\mathrm{Shuffle}} = \left\{ (s, w) \;\middle|\; \begin{array}{l} s = ([m_1], \ldots, [m_\tau], \hat{m}_1, \ldots, \hat{m}_\tau, \hat{m}_i \in R_p^*, \\ w = (\gamma, f_1, \ldots, f_\tau, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_\tau), \gamma \in S_\tau, \\ \forall i \in [\tau] : \ \mathtt{Open}([m_{\gamma^{-1}(i)}], \hat{m}_i, \boldsymbol{r}_i, f_i) = 1 \end{array} \right\}$$

To use an idea similar to Neff, we would need that all messages $\hat{m}_i$ are invertible. As this may not be the case for arbitrary ring elements, we restrict the message space to invertible elements. The first step for our shuffle protocol will be that $\mathcal{V}$ picks a random invertible element ρ not in $\{\hat{m}_i\}$ and sends ρ to $\mathcal{P}$. $\mathcal{P}$ and $\mathcal{V}$ then compute the values $\hat{M}_i, M_i$ by setting $M_i = m_i - \rho$, $\hat{M}_i = \hat{m}_i - \rho$. The proof, on a high level, then shows that $\prod_i M_i = \prod_i \hat{M}_i$. This is in fact sufficient, as the $m_i, \hat{m}_i$ can be considered as roots of polynomials of degree τ. By subtracting ρ from each such entry and multiplying the results we obtain the evaluation of these implicit polynomials in the point ρ, and if the $\hat{m}_i$ are not a permutation of the m_i then these implicit polynomials will be different. At the same time, the number of points on which both polynomials can agree is upper-bounded by τ^δ / p^N, which is negligible for all relevant parameters.

Our public-coin zero-knowledge protocol proves this identity of evaluations of these two polynomials by showing that a particular set of linear relations (2) is satisfied (where we show in the formal proof how it is related to the aforementioned product of $M_i, \hat{M}_i$).

Zero-Knowledge Proof of Correct Shuffle

$$
\begin{array}{lcl}
\underline{\text{Prover}} & & \underline{\text{Verifier}} \\
 & \xleftarrow{\quad \rho \quad} & \rho \overset{\$}{\leftarrow} R_p^* \\
\hat{M}_i = \hat{m}_i - \rho & & \hat{M}_i = \hat{m}_i - \rho \\
M_i = m_i - \rho & & [M_i] = [m_i] - \rho \\
\theta_i \overset{\$}{\leftarrow} R_p, \theta_0 = \theta_\tau = 0 & & \\
[D_i] = [\theta_{i-1} M_i + \theta_i \hat{M}_i] & \xrightarrow{\{[D_i]\}_{i=1}^{\tau}} & \\
 & \xleftarrow{\quad s_0 = \beta \quad} & \beta \overset{\$}{\leftarrow} R_p \\
s_1 = \theta_1 - \beta \dfrac{M_1}{\hat{M}_1} & & \\
s_i = \theta_i + \theta_{i-1} \dfrac{M_i}{\hat{M}_i} - s_{i-1} \dfrac{M_i}{\hat{M}_i} & & \\
s_{\tau-1} = \theta_{\tau-1} \dfrac{M_\tau}{M_{\tau-1}} + (-1)^{\tau-1} \beta \dfrac{\hat{M}_\tau}{M_{\tau-1}} & & \\
\pi_{L_i} \leftarrow \Pi_{\mathrm{Lin}}(([M]_i, \hat{M}_i), D_i, (s_{i-1}, s_i)) & \xrightarrow{\{s_i\}_{i=1}^{\tau-1}, \{\pi_{L_i}\}_{i=1}^{\tau-1}} & \Pi_{\mathrm{LinV}}
\end{array}
$$

Fig. 6: The public-coin zero-knowledge protocol of correct shuffle.

As a first step, $\mathcal{P}$ draws $\theta_i \overset{\$}{\leftarrow} R_p$ uniformly at random for each $i \in \{1, \dots, \tau\}$, and computes the commitments

$$
\begin{aligned}
[D_1] &= \left[\theta_1 \hat{M}_1\right] \\
\forall j \in \{2, \dots, \tau-1\}: \ [D_j] &= \left[\theta_{j-1} M_j + \theta_j \hat{M}_j\right] \\
[D_\tau] &= [\theta_{\tau-1} M_\tau] \, .
\end{aligned}
\tag{1}
$$

$\mathcal{P}$ then sends these commitments $\{[D_i]\}_{i=1}^{\tau}$ to the verifier $\mathcal{V}$, which in turn chooses a challenge $\beta \in R_p$, whereupon $\mathcal{P}$ computes $s_i \in R_q$ such that the following equations are satisfied:

$$
\begin{aligned}
\beta M_1 + s_1 \hat{M}_1 &= \theta_1 \hat{M}_1 \\
\forall j \in \{2, \dots, \tau-1\}: \ s_{j-1} M_j + s_j \hat{M}_j &= \theta_{j-1} M_j + \theta_j \hat{M}_j \\
s_{\tau-1} M_\tau + (-1)^\tau \beta \hat{M}_\tau &= \theta_{\tau-1} M_\tau .
\end{aligned}
\tag{2}
$$

To verify the relations, $\mathcal{P}$ uses the protocol Π_{Lin} from Section 3 showing that the content of each commitment $[D_i]$ is such that D_i, M_i and $\hat{M}_i$ are as defined in (2). The protocol ends when $\mathcal{V}$ has verified all the τ linear equations in (2). In order to compute the s_i values, we can use the following fact:

Lemma 1. *Choosing*

$$s_j = (-1)^j \cdot \beta \prod_{i=1}^{j} \frac{M_i}{\hat{M}_i} + \theta_j \tag{3}$$

for all $j \in 1, \ldots, \tau - 1$ yields a valid assignment for Equation (2).

□

From this Lemma it is clear that the protocol is indeed complete. Interestingly, this choice of s_j also makes these values appear random: each s_j is formed by adding a fixed term to a uniformly random private value θ_j. This will be crucial to show the zero-knowledge property formally. The following statement can be shown about Π_{Shuffle}:

Theorem 5. *Assume that* $(\texttt{KeyGen}_{\text{C}}, \texttt{Com}, \texttt{Open})$ *are a secure commitment scheme with Π_{Lin} as a HVZK Proof of Knowledge of the relation $\mathcal{R}_{\text{Lin}}$. Furthermore, assume that $\tau^\delta/|\Sigma_{\beta_\infty}|+1/p^N < 2^{-\aleph}$. Then the protocol Π_{Shuffle} is an HVZK Proof of Knowledge for the relation $\mathcal{R}_{\text{Shuffle}}$ with soundness error $2^{-\aleph}$.*

□

C Size and Timings of the Shuffle Protocol

C.1 Parameters and Size

Parameter	Explanation	Constraints
N, δ	Degree of polynomial $X^N + 1$ in R	$N \geq \delta \geq 1$, where N, δ powers of two
p	Modulus for commitments	Prime p such that $p = 2\delta + 1 \mod 4\delta$
σ_{C}	Standard deviation of discrete Gaussians	Chosen to be $\sigma_{\text{C}} = 11 \cdot \nu \cdot \beta_\infty \cdot \sqrt{kN}$
k	Width (over R_p) of commitment matrix	
n	Height (over R_p) of commitment matrix	
ν	Maximum l_1-norm of elements in $\mathcal{C}$	
$\mathcal{C}$	Challenge space	$\mathcal{C} = \left\{c \in R_p \mid \|c\|_\infty = 1, \|c\|_1 = \nu\right\}$
$\bar{\mathcal{C}}$	The set of differences $\mathcal{C} - \mathcal{C}$ excluding 0	$\bar{\mathcal{C}} = \{c - c' \mid c \neq c' \in \mathcal{C}\}$

Table 2: Table of parameters for our scheme.

Parameters. To determine secure while not enormously big parameters for our scheme, we need to first make sure that we have sufficiently large parameters to ensure both binding and hiding of the commitments for which we will use the "optimal" parameter set of [4] which is both computationally binding and hiding, see Table 3.

Parameter	Commitment
N	1024
p	$\approx 2^{32}$
σ	$\sigma_{\mathrm{C}} \approx 46000$
ν	36
δ	2
n	1
k	3
Proof	6.6 KB
Primitive	8.1 KB

Table 3: Parameters for the commitments in Baum et al. [4].

Size of the Shuffle Proof. Our shuffle protocol is a $4+3\tau$-move protocol with elements from R_p. Each element in R_p has at most N coefficients of size at most p, and hence, each R_p-element has size at most $N \log p$ bits. For every R_p-vector that follows a Gaussian distribution with standard deviation σ we assume that we can represent the element using $N \cdot \log(6\sigma)$ bits. Every element from $\mathcal{C}$ will be assumed to be representable using at most $2N$ bits.

We analyze how much data we have to include in each step of the shuffling protocol in Figure 6. Using the Fiat-Shamir transform [19] we can ignore the challenges from the verifier. The prover ends up sending 1 commitment, 1 ring-element and 1 proof of linearity per messages. Using the parameters from Table 3, we get a shuffle proof of total size $\approx 19\tau$ KB.

C.2 Efficiency

We collected performance figures collected from our prototype implementation written in C to estimate the runtime of our scheme. Estimates are based on Table 3 and the implementation was benchmarked on an Intel Skylake Core i7-6700K CPU running at 4GHz without TurboBoost. Timings are available in Table 4, and we plan to publish our code together with the final version of the paper.

Our Scheme:	Commit	Open	Shuffle
Time	0.9 ms	1.2 ms	18τ ms

Table 4: Timings for cryptographic operations.

Elementary Operations. Multiplication in R_p is usually implemented when $p \equiv 1 \bmod (2N)$ and $X^N + 1$ splits in N linear factors, for which the Number-Theoretic Transform is available. Unfortunately, Baum et al. [4] restricts the parameters and we instead adopt $p \equiv q \equiv 5 \bmod 8$. In this case, $X^N + 1$ splits in two $N/2$-degree irreducible polynomials $(X^{N/2} \pm r)$ for r a modular square root of -1. This gives an efficient representation for $a = a_1 X^{N/2} + a_0$ using the Chinese Remainder Theorem: $CRT(a) = (a \pmod{X^{N/2} - r}, a \pmod{X^{N/2} + r})$. Even though the conversions are efficient due to the choice of polynomials, we sample ring element directly in this representation whenever possible. As in [27], we implement the base case for degree $N/2$ using the FLINT arithmetic library for polynomials [21].

Commitment. A commitment is generated by multiplying the matrix $\boldsymbol{A}$ by a vector $\boldsymbol{r}_m$ and finally adding the message m to the second component in the CRT domain. Computing and opening a commitment takes 0.9 ms and 1.2 ms, respectively, and sampling randomness $\boldsymbol{r}_m$ was excluded from this measurement.

Shuffle Proof. Benchmarking includes all samplings required inside the protocol and amounts to 18τ ms for the entire proof.

D Subliminal *l*-Free Digital Signatures

Definition 10 (Subliminal l-Free Digital Signatures). *Let m be a fixed message, $\mathcal{S}$ a signature scheme, $\hat{m}$ a subliminal message chosen by* S *and s some secret pre-shared information between* S *and* 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* W *will close the channel if* S *deviates from the protocol. Then we say that $\mathcal{S}$ is a subliminal l-free digital signature scheme if* S *cannot reliably both create a proof π accepted by* W *and at the same time encode a subliminal message $\hat{m}$ of size greater or equal to l into σ decodable by* R*, but cannot be decoded nor detected by* W *unless* S *does work exponentially in l.*

E Existential Unforgeability for Subliminal-Free Signatures

Definition 11 (Existential Unforgeability). *We say that a subliminal-free digital signature scheme is* existential unforgeable *if an adversary* A*, after given valid signatures σ_i and proofs π_i of messages m_i of* 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* A *to be:*

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

F 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, we don't need to give a proof of correct sum but can sum together all the commitments directly), 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 large 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 [1], denoted $\texttt{GaussSampler}_{\sigma}$. Choosing a binary string ρ of length 3λ, the values s_i can be generated as $s_i \leftarrow \texttt{GaussSampler}_{\sigma}(\rho, i)$. 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 $\texttt{H}(j, \hat{\rho}, \{\tilde{c}_i\}_{i=1}^{\tau}, l)$ for $1 \leq j \leq 3$, $1 \leq l \leq \kappa$, and a suitable hash function $\texttt{H}$. Here, $\hat{\rho} = \rho$ in the non-interactive case, while $\hat{\rho}$ is a random string of size 3λ 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 [18]. 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 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_i} 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. [4] 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 Appendix C we estimate that it takes 1τ ms $= 0.256$ s to create the commitments, 18τ ms ≈ 4.6 s to generate the shuffle proof and 1.9τ ms ≈ 0.5 s to verify a shuffle proof. In summary, it takes approximately 5 seconds to generate subliminal-free signatures, and each signature is of size 12.65 MB if we want $\lambda = 128$ bit security. We can use the efficient lattice-based signature schemes like Dilithium [17] or qTesla [2] for the signature, where both schemes produce signatures of size 2.7 KB. However, the signatures are small compared to the overhead of the VRS.

The hash-based SFS. This VRS construction works with any Schnorr-like signature scheme, but we'll instantiate the signatures using the lattice-based signature schemes Dilithium [17] and qTesla [2] 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 exponential in the security parameter 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, which is then exponential. 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 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 $w_i = \mathtt{OWF}(v_i)$. Let $|\mathtt{H}|$, $|\mathtt{Sample}|$ and $|\mathtt{OWF}|$ denote the cost it takes to compute a hash, the sampling and the one-way function, respectively. The total computation is then $T = 3M|\mathtt{H}| + M(|\mathtt{Sample}| + |\mathtt{OWF}|)$. Verification is similar.

The size of the communication is relatively small. The commitment $\tilde{c}$ is the output of the hash function $\mathtt{H}$ of size 3λ, the commitment c is the output from $\mathtt{OWF}$, and the proof π_I contains $\log M$ hash values of total size $3\lambda \log_2 M$ bits.

We chose $\lambda = 128$ bits of security for our signature scheme, but allow for a soundness error $1/M$ where $M_1 = 2^{10}$, $M_2 = 2^{15}$ or $M_3 = 2^{20}$ in our SFS. We chose the $\mathtt{SHA}-\mathtt{384}$ hash function. By running the command `openssl speed sha512` in the locally (openssl doesn't offer timings for $\mathtt{SHA}-\mathtt{384}$, but this should give us an accurate estimate), we estimate that we can compute roughly 2^{23} $\mathtt{SHA}-\mathtt{512}$-hashes per second with messages of size 512 bits as input (most hashes are computed on short messages). We also assume that we can compute the lattice-based commitments in roughly 1 ms, as above, which includes both $\mathtt{OWF}$ and $\mathtt{Sample}$. qTelsa give the same estimate for computing a signature. The prover and verifier time is:

$$T_1 \approx 3 \cdot 2^{10} \frac{1}{2^{23}} + 2^{10} \frac{1}{1000} \approx 1 \text{ s},$$
$$T_2 \approx 3 \cdot 2^{15} \frac{1}{2^{23}} + 2^{15} \frac{1}{1000} \approx 33 \text{ s},$$

$$T_3 \approx 3 \cdot 2^{20} \frac{1}{2^{23}} + 2^{20} \frac{1}{1000} \approx 18 \text{ min}.$$

Next, ignoring the signature size, the total proof size is $S = 3\lambda(1 + \log_2 M)$:

$$S_1 = 0.528 \text{ KB}, S_2 = 0.768 \text{ KB}, S_3 = 1.008 \text{ KB}.$$

Using the signature schemes Dilithium or qTesla both give us subliminal-free signatures a total size $\approx 3.3 - 3.7$ KB, only a factor $1.19 - 1.37$ times larger than the subliminal versions of the signature schemes, depending on M.

Finally, we note that the randomness created in the VRS, might not pass the rejection sampling in the signature scheme. For Dilithium and qTesla we expect up to three trials to create a signature, and hence, we may allow a sender to run the VRS up to three times with abort before sending a signature. This allows for a subliminal 1-bit channel with 87.5% likelihood of success, which would be considered somewhat reliable. If the signer abort too many times, warden closes the channel completely.