Subliminal channels in post-quantum digital signature schemes

Herman Galteland* and Kristian Gjøsteen

Department of Mathematical Sciences,
NTNU – Norwegian University of Science and Technology, Trondheim
{herman.galteland, kristian.gjosteen}@ntnu.no

May 27, 2019

Abstract

We analyze the digital signatures schemes submitted to NIST's Post-Quantum Cryptography Standardization Project in search for subliminal channels.

Keywords. Subliminal channels, post-quantum, digital signatures.

1 Introduction

The *NIST Post-Quantum Cryptography Standardization Project* (PQCSP) [18] is analyzing and testing post quantum secure public key encryption, key encapsulation mechanism, and digital signature schemes submitted by the cryptographic research community. NIST's goal is to have a selection of standardized schemes for future use.

In this paper we analyze the proposed digital signature schemes submitted to PQCSP for *subliminal channels*, covert communication channels that uses existing cryptographic protocols to send information, called *subliminal messages*, the protocols were not intended for.

Knowledge of subliminal channels is important for those who want to use them and for those who want avoid them. Privacy concerned users could use subliminal channels to bypass censorship or avoid surveillance. When constructing secure systems developers might avoid schemes with subliminal channels to prevent, for example, information leakage or malicious actors secretly breaching their system. It is likely that some of the digital signatures schemes submitted to PQCSP will be used. It is therefore useful to know about any subliminal channels present in these schemes.

We use the following model when we look for subliminal channels, where we try to fulfill the goals of the sender and receiver of the subliminal messages.

Subliminal channel model The *subliminal sender* wants to communicate discretely with a *subliminal receiver* over a channel controlled by an adversary. We shall assume that the subliminal sender is sending message-signature pairs that the subliminal receiver later can observe.

Hiding information directly in the message is steganography, which is not our current topic of interest. Therefore we shall assume that the message part of the message-signature pairs will not be under the subliminal senders' control. Instead, the subliminal sender generates the signatures and will try to embed subliminal messages in the signatures.

We assume that the subliminal sender and the subliminal receiver share a key for a suitable symmetric cryptosystem (that may even be stateful). In particular, this means that the subliminal message to be embedded will be indistinguishable from random bits.

We may allow the subliminal receiver to know the signing key, but it is better if the receiver does not need the signing key. The subliminal sender may also be the one generating the key, and in this case we may allow cheating during key generation.

*This work is funded by Nasjonal sikkerhetsmyndighet (NSM), www.nsm.stat.no

The subliminal sender's goal is to send as much information as possible without detection. That is, except for the subliminal receiver, anyone who inspects the generated signatures should not be able to decide if they contain subliminal messages or were generated honestly. This should hold even if the one who inspects the signatures also chooses the messages to be signed.

We note that even when the subliminal channel is used, the signature scheme should remain secure in the ordinary sense. (It may seem like this follows from indistinguishability, since if the scheme is insecure when the subliminal channel is used, a distinguisher would try to break the scheme and distinguish in that way. However, since checking for subliminal channels is something that we want to do regularly, any subliminal channel distinguisher must be fast. Which means that it may not have time to run an attack against the signature scheme.)

1.1 Examples of subliminal channels

One would expect a deterministic signature scheme to be free from subliminal channels. Signatures of the RSA-FDH scheme [7] have the form $\text{Sign}_{RSA-FDH} = (H_{FDH}(M))^d \mod n$, for an RSA decryption key d and modulus n. There are no random value to exploit, however, it is possible to use it to send (short) subliminal messages. Using the halting strategy [21] we can in some sense make a 1 bit subliminal channel. However, this is a generic attack.

Any sufficiently non-deterministic signature scheme allows a generic subliminal channel. For example, to send n bits, generate signatures until the first n bits of the signature matches the message. We can generalize the approach using the leftover hash lemma [34], where we generate signatures until the hash of the signature matches the message to be sent. This holds for any signature scheme. Of course, the computational cost of this approach is exponential in n, so this will be a low-bandwidth channel.

It is possible to *derandomize* signature schemes by replacing the random bits with the output of a pseudo-random function applied to the message. In general this will not prevent subliminal channels [12]. In most construction, it is hard to verify that the signature was deterministically generated without knowledge of the secret key. (Another option is to sign the same message twice. We assume the subliminal sender keeps track of messages sent.) It is, however, possible to check if there was a subliminal channel present assuming that the secret value becomes public at a later point.

The PSS message encoding scheme, used with the RSA or Rabin primitive to make a signature scheme [7], has a subliminal channel that only requires *public* values to recover the message. Taking RSA as an example, we have $\text{Sign}_{PSS-RSA} = (0||w||r^*||M)^d \mod n$, for a RSA private key d, modulus n, hash value w, random value r^*, and message M. Using the RSA public key e we can recover the randomness r^*. Replacing the random value r^* with a subliminal message m_s we can make a subliminal channel. The Rabin primitive works in the same way.

A subliminal channel using Schnorr Signatures [47] requires the *secret* values to recover the message, which has to be shared with the subliminal receiver. A Schnorr signature has the form $\text{Sign}_{Schnorr} = (y, e) = (r + se, e)$, where s is the secret key, e is a hash value, and r is a random value. We can use s to recover r by computing $r = y - se$. It is then trivial to replace the randomness with a subliminal message m_s to make a subliminal channel.

1.2 Our contributions

We modify the proposed digital signature schemes submitted to PQCSP to reliably send subliminal messages, typically by exploiting any random values, while keeping the signatures valid. We show how to insert and recover subliminal messages and give an overview of the subliminal bandwidth of each channel found.

1.3 Related work

Simmons motivated his work on the subliminal channel with the prisoners' problem [48], where two prisoners wish to plan their escape and the warden allows them to send signed message if he can read the content of the messages. The problem for the prisoners is to communicate covertly using their monitored communication channel, to plan their escape. The problem of the warden is to discover and prevent the existence of subliminal channels, to prevent prisoners escaping. Using digital signatures Simmons showed that such

channels exist [49–51], and since then more have been found in various digital signature schemes [4, 12, 36, 54]. Hartl et al. showed that it is possible to insert subliminal messages in signature schemes based on the multivariate quadratic polynomial problem [30].

Desmedt made the first attempt of making a subliminal-free authentication and signature scheme [20], and since then more schemes have been constructed to prevent subliminal channels [11, 12, 24, 51, 52]. Divertible protocols [10, 14, 15, 41] are separable by an unnoticeable third party (a warden) sitting in between two communicating principals (the prisoners), where the third party can rerandomize messages to remove any subliminal channel. Cryptographic reverse firewalls [16, 40] can, in theory, be used to prevent subliminal channels and existing constructions are based on Decisional Diffie–Hellman and use rerandomization techniques to remove subliminal messages, which are not suitable for post-quantum secure schemes.

1.4 Notation

We try to follow the notation given in each signature scheme as much as we can, where we describe each scheme's notation when they are introduced. We denote the subliminal message in plain m_s or in bold $\mathbf{m}_s$ to fit with the notation of each scheme. Let $\text{sgn}(x)$ denote the sign of x and is equal to 1 if $x \geq 0$ or equal to -1 if $x < 0$. When the distribution of random elements are not specified, other than that they are random, we assume it is uniform.

1.5 Overview

In Section 2 we briefly describe each scheme submitted to NIST's Post-Quantum Cryptography Standardization Project and show the subliminal channel(s) we have found. In Section 3 we summarize and give a table of each channel's subliminal bandwidth.

2 Proposed digital signature schemes

2.1 Ideas

While we analyze many different schemes, many ideas are reused for many different schemes and we discuss some of the theory here.

In addition to the *public* and *secret* channels described in Section 1.1 we also note the following subliminal channels. Some scheme use random values that are *included in the clear*, which can be used to embed encrypted subliminal messages. A few lattice based schemes use a large random value to hide a small secret value, where we can insert information in the *higher order digits* of the random value.

Some signature schemes require the signatures to have a specific form. By using nondeterministic encryption we can simply encrypt the subliminal messages many times until we get a signature of the desired form. This will reduce the available bandwidth, but the cost is usually just a few bits, especially if stateful encryption is used.

The random elements used in the signature scheme are sampled according to some distribution. For non-uniform distributions we can sometimes do rejection sampling on the randomness used to encrypt the subliminal message such that the output is close to the desired distribution.

2.2 CRYSTALS – Dilithium

The Dilithium signature scheme is based on Fiat-Shamir with Aborts [37] where the security is based on the shortest vector problem. The scheme uses the polynomial rings $\mathcal{R} = \mathbb{Z}[X]/\langle X^n+1\rangle$ and $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n+1\rangle$, for an integer n. An element $x \in \mathcal{R}$, and $x \in \mathcal{R}_q$, is denoted in plain, and vectors and matrices are denoted in bold. An element in $B_{60} \subset \mathcal{R}$ has 60 coefficients that are either -1 or 1 and the rest are 0.

Signatures have the form

$$\text{Sign}_{Dilithium} = (\mathbf{z}, \mathbf{h}, c),$$

where $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$, $\mathbf{y} \in \mathcal{R}_q^l$ is uniformly random, $\mathbf{s}_1 \in \mathcal{R}_q^l$ is part of the secret key, $\mathbf{h}$ is a Boolean vector used to recover high-order bits, and $c \in B_{60}$ is the hash of the message digest and the higher-order bits of a public matrix multiplied by the vector $\mathbf{y}$. The vector $\mathbf{z}$ has max norm $\|\mathbf{z}\|_\infty \leq \gamma_1 - \beta - 1$.

Table 1: CRYSTALS – Dilithium parameters and subliminal message bounds for high order digits subliminal channel.

		Parameter set			
Parameter	Description	I	II	III	IV
n	degree	256	256	256	256
l	dimension	2	3	4	5
γ_1	randomness bound	523776	523776	523776	523776
β	bound	375	325	275	175
$\|\mathbf{m}_s\|_\infty$	message bound	523	523	523	523
$\|r'_i\|$	masking value	$[375, 624]$	$[325, 674]$	$[275, 724]$	$[175, 824]$

2.2.1 Subliminal channel using secret values

Assume that the sender and receiver both know the secret key $sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$. The sender can replace the random value $\mathbf{y}$ with a subliminal message $\mathbf{m}_s$, that is, let $\text{Sign}_{Dilithium} = (\mathbf{z}, \mathbf{h}, c)$, where $\mathbf{z} = \mathbf{m}_s + c\mathbf{s}_1$. With the secret key the receiver can retrieve the subliminal message as $\mathbf{m}_s = \mathbf{z} - c\mathbf{s}_1$. We can re-encrypt the subliminal messages if the produced signature is not a valid signature.

2.2.2 Subliminal channel using high order digits

The scheme requires that $\|c\mathbf{s}_1\|_\infty \leq \beta$ and $\|\mathbf{z}\|_\infty \leq \gamma_1 - \beta - 1$. We can insert a subliminal message in the high order digits of $\mathbf{y}$ if β is small compared to the coefficients of $\mathbf{y}$.

Proof of concept For the proposed parameters, the coefficients of the random value have at most six digits and the coefficients of the error term $\|c\mathbf{s}_1\|_\infty$ have at most three digits. By bounding the randomness $\|\mathbf{y}\|_\infty \leq \gamma_1 - 2\beta - 1$ we should be able to send subliminal messages with three digit coefficients. Let $\mathbf{y} = 1000\mathbf{m}_s + \mathbf{r}'$ and

$$\mathbf{z}' = 1000\mathbf{m}_s + \mathbf{r}' + c\mathbf{s}_1,$$

where $\mathbf{m}_s \in \mathcal{R}_q^l$ and $\mathbf{r}' \in \mathcal{R}_q^l$. The bounds on the coefficients of $\mathbf{m}_s$ and the coefficient interval of the masking randomness $\mathbf{r}'$ are in Table 1. Note that each coefficient of $\mathbf{r}'$ need to have the same sign as the subliminal message, $\text{sgn}(r'_i) = \text{sgn}(m_{s_i})$ for all $i \in \{1, \ldots, n\}$, for the message recovery to be correct. The subliminal message $\mathbf{m}_s$ is bounded such that $\|\mathbf{z}'\|_\infty \leq \gamma_1 - \beta - 1$, and the coefficients of the masking value $\mathbf{r}'$ are chosen such that $0 \leq \|\mathbf{r}' + c\mathbf{s}_1\|_\infty \leq 999$. The receiver computes

$$\left\lfloor \frac{\mathbf{z}'}{1000} \right\rfloor$$

to recover the subliminal message $\mathbf{m}_s$. When we encrypt the subliminal message we can re-encrypt it if the produced signature does not meet the requirements of the signature algorithm. We should also pick the masking values $\mathbf{r}'$ such that $\|\mathbf{z}'\|_\infty \leq \gamma_1 - \beta - 1$.

2.3 DME

The DME signature scheme is based on multivariate polynomials. The scheme uses the field $\mathbb{F}_p$ where vectors are denoted in plain lowercase.

Signatures have the form

$$\text{Sign}_{DME} = (x, z_0, h_1),$$

where $z_0 \in \mathbb{F}_p^{N_1}$ is a message, $h_1 : \{1, \ldots, N_1\} \to \{1, \ldots, e \cdot n \cdot m\}$ is a map used to add a random padding to the messages, and $x = F^{-1}(z_0) \in \mathbb{F}_p^{N_1}$. If z_0 is not in the image of F, pad z_0, using h_1, to get $z \in \text{Im}(F)$ and set $x = F^{-1}(z)$. The values N_1, e, n, and m are parameters. A verifier computes $z = F(x)$, discards the padding to get z'_0, and verifies that $z'_0 = z_0$.

2.3.1 Subliminal channel using public values

Anyone with the public key can recover $z = F(x)$ and, specifically, the random padding which can be replaced with a subliminal message. The scheme offer two parameter sets where up to 16 and 32 bits are added to messages, respectively. If the padded message z is not in the image of F, for a chosen padding, we can re-encrypt the subliminal message until we find a z which is.

2.4 DRS

The DRS signature scheme is a variation of the scheme by Plantard et al. [45], where both schemes are based on GGH [29]. The security of the scheme is based on the guaranteed distance decoding problem. The scheme uses a diagonal dominant lattice $\mathcal{L}(P)$, with public basis P that has large coefficients. An element $x \in \mathcal{L}(P)$ is an integer vector.

Signatures have the form

$$\text{Sign}_{DRS} = (k, v, w),$$

where k satisfies $kP = v - w$, $v \in \mathbb{Z}^n$ is the hash of the message, and $w \in \mathbb{Z}^n$ is a reduced vector of v that satisfies $w \equiv v \mod \mathcal{L}(S)$ and $\|w\|_\infty < D$. The public basis P has big coefficients and is constructed from the secret basis S using unimodular matrices U, and D is a max norm bound.

2.4.1 No channel found

We were unable to find a subliminal channel in the DRS signature scheme.

2.5 DualModeMS

The DualModeMS scheme has two layers, an inner and an outer. The inner layer is a Matsumoto-Imai multivariate construction [38] based on FHEv, and the outer layer modifies the output of the inner layer by using the method of Szepieniec, Beullens, and Preneel [53]. The scheme uses the fields $\mathbb{F}_2$ and $\mathbb{F}_2^n$, where vectors are denoted in bold lower case, matrices are denoted in bold uppercase and polynomials are denoted in capital plain letters.

The output of the *inner* algorithm is

$$\mathbf{s} = (\phi(Z), \mathbf{v}) \times \mathbf{S}^{-1},$$

where $\phi(Z)$ is a root represented as a binary vector, $\mathbf{S}$ is an invertible $n + \nu \times n + \nu$ matrix, and $\mathbf{v} \in \mathbb{F}_2^\nu$ are the random vinegar variables. A signature of the DualModeMS scheme has the form

$$\text{Sign}_{DualModeMS} = (\mathbf{s}_1, \ldots, \mathbf{s}_\sigma, \mathbf{h}, \texttt{openpaths}),$$

where the $\mathbf{s}_i$'s are output of the inner algorithm, $\mathbf{h}$ is a random set of linear combination of the public key, and `openpaths` is a set of Merkle tree paths used to compute the public key.

2.5.1 Subliminal channel using public values

For a chosen vector $\mathbf{s}'$, that contains a subliminal message in the last ν bits, we can find a vector $\mathbf{v} = \mathbf{s}' \times \mathbf{S}$ such that $\mathbf{s} = (\phi(Z), \mathbf{v}) \times \mathbf{S}^{-1}$ contains an encrypted subliminal message in the last ν bits. If $\mathbf{s}'$ is not in the image of $\mathbf{S}^{-1}$ we can re-encrypt the subliminal message.

The subliminal message $\mathbf{m}_i \in \mathbb{F}_2^\nu$ is inserted in the last ν bits of $\mathbf{s}_i \in \mathbb{F}_2^{n+\nu}$, for $i = 1, \ldots, \sigma$. The scheme provides three parameter sets. The dimension ν is 11, 18, and 32, respectively. The number of inner signatures σ is 64, 96, and 256, respectively. If, for a chosen subliminal message, the inner algorithm does not terminate we can re-encrypt the subliminal message and try again.

2.6 Falcon

The Falcon signature scheme is a combination of the GPV framework [26], over the NTRU lattice, with fast Fourier sampling. The scheme uses the polynomial ring $\mathbb{Z}[x]/\langle\phi\rangle$, where $\phi \in \mathbb{Z}[x]$ is a monic and irreducible cyclotomic polynomial of degree n. Vectors are denoted in lowercase bold and matrices in uppercase bold.

Signatures have the form

$$\text{Sign}_{Falcon} = (\mathbf{r}, \mathbf{s}),$$

where $\mathbf{r} \in \{0,1\}^{320}$ is a uniformly sampled salt, and is used in a hash together with the message to get a point $c \in \mathbb{Z}_q[x]/\langle\phi\rangle$. A preimage $\mathbf{t} = c\mathbf{B}^{-1}$ is computed, for a secret basis $\mathbf{B}$, and is used to find two short polynomials $s_1, s_2 \in \mathbb{Z}_q[x]/\langle\phi\rangle$ such that $s_1 + s_2 h \equiv c \mod q$, for a NTRU public key $h = gf^{-1} \mod q$. The polynomial s_2 is encoded as the bitstring $\mathbf{s}$.

2.6.1 Random values included in the clear

The random value $\mathbf{r} \in \{0,1\}^{320}$ can be replaced by a subliminal message $\mathbf{m}_s \in \{0,1\}^{320}$.

2.7 G*e*MSS

The GeMSS scheme is based on the `Quartz` digital signature scheme [43], modified using the ideas of the `Gui` digital signature scheme [23]. The scheme uses the fields $\mathbb{F}_2$ and $\mathbb{F}_2^n$. Vectors are denoted in bold lowercase, matrices in bold uppercase and polynomials in capital plain letters.

The signature algorithm of the GeMSS scheme uses a subroutine called GeMSS inversion, which output the $n + \nu$ bit vector

$$\mathbf{s} = (\phi(Z), \mathbf{v}) \times \mathbf{S}^{-1},$$

where $\phi(Z)$ is a root represented as a binary coefficient vector, $\mathbf{S}$ is a secret invertible $n + \nu \times n + \nu$ matrix, and $\mathbf{v} \in \mathbb{F}_2^\nu$ are random vinegar variables. A signature of GeMSS has the form

$$\text{Sign}_{GeMSS} = (\mathbf{S}_{nb\_ite}, \mathbf{X}_{nb\_ite}, \ldots, \mathbf{X}_1),$$

where $(\mathbf{S}_i, \mathbf{X}_i)$ is the output $\mathbf{s}_i$ of the inversion algorithm discussed above. The first component $\mathbf{S}_i \in \mathbb{F}_2^m$ is the first m bits of the output $\mathbf{s}_i$ and the second component $\mathbf{X}_i \in \mathbb{F}_2^{n+\nu-m}$ is the remaining $n + \nu - m$ bits. The last ν bits of $\mathbf{s}_i$ is contained in the second component $\mathbf{X}_i$.

2.7.1 Subliminal channel using public values

For a chosen vector $\mathbf{s}'$, that contains a subliminal message in the last ν bits, we can find a vector $\mathbf{v} = \mathbf{s}' \times \mathbf{S}$ such that $\mathbf{s} = (\phi(Z), \mathbf{v}) \times \mathbf{S}^{-1}$ contains an encrypted subliminal message in the last ν bits. If $\mathbf{s}'$ is not in the image of $\mathbf{S}^{-1}$ we can re-encrypt the subliminal message.

The subliminal message $\mathbf{m}_i \in \mathbb{F}_2^\nu$ is inserted in the last ν bits of $\mathbf{s}_i \in \mathbb{F}_2^{n+\nu}$, for $i = 1, \ldots, nb\_ite$. The scheme provides three parameter sets. The dimension ν is 12, 20, and 33, respectively, and the number nb_ite is 4 for all sets. If the inversion algorithm does not terminate, for a chosen subliminal message, we can re-encrypt the message and try again.

2.8 Gravity – SPHINCS

Gravity–SPHINCS is an extension of Goldreich's construction of a stateless hash based signature scheme [28], and share many similarities with SPHINCS [8]. The scheme outputs bit strings, where a binary number $v \in \{0,1\}^n$ is denoted in plain. The scheme uses four types of trees; hyper tree, subtrees (Merkle trees [39]), WOTS public key compression trees, and PORST public key compression trees. The signature is composed of a PORST signature (improved version of HORST few times signatures [8]), Winternitz one time signatures [33], and Merkle authentication paths.

Signatures have the form

$$\text{Sign}_{Gravity-SPHINCS} = (s, \sigma_d, oct, \sigma_{d-1}, A_{d-1}, \ldots, \sigma_0, A_0, A_c),$$

where s is a hash and a public salt, σ_d is a PORST signature, oct is an authentication value for the PORST signatures, $\sigma_{d-1}, \ldots, \sigma_0$ are Winternitz signatures, and $A_{d-1}, \ldots, A_0, A_c$ are Merkle authentication paths.

2.8.1 Random values included in the clear

The public salt s is constructed using a hash function with a secret salt and the message as input, where the verifier cannot verify that the random value s was computed using this hash function without the secret salt. The sender can replace the random value s with a subliminal message m_s.

The PORST signature $\sigma_d = (s_{x_1}, s_{x_2}, \ldots, s_{x_k})$, where the x_i's are indices and the s_i are generated using a pseudorandom function G with a secret seed and an address in the hyper tree as input. We can replace the random vales with a subliminal message. The security of the scheme will be reduced as we can now make a collision in the PORST signatures with probability 2^{-128}, which is acceptable.

The submission proposes tree parameter sets. The binary string of length n is 256 for all parameter sets, and the dimension k is 24, 32, and 28, respectively.

2.9 Gui

The Gui scheme is based on the Hidden Field Equations cryptosystem using the minus and vinegar modification (`HFEv-`) [43]. The scheme use a finite field $\mathbb{F}_q$ and the field extension $\mathbb{F}_{q^n}$. Field elements are denoted in lowercase plain. Vectors are denoted in lowercase bold or in uppercase plain. The affine transformations and maps are denoted in uppercase plain and their inverses are denoted with the prefix Inv.

The signature algorithm of the Gui scheme uses a subroutine called HFEv- inversion that outputs

$$\mathbf{z} = InvT \cdot ((\mathbf{y}||v_1|| \ldots ||v_\nu) - c_T),$$

where $invT$ is the inverse of the affine transformation $\mathcal{T}$, $\mathbf{y}$ is a root of a polynomial Y, $v_1, \ldots, v_\nu$ are random vinegar elements, and c_T is a random masking value. The output of the signature scheme is

$$\text{Sign}_{Gui} = (S_k||X_k|| \ldots ||X_1||r),$$

where r is a random bit string and (S_i, X_i) is the output of the HFEv- inversion algorithm. S_i is the first $n - a$ elements and X_i is the last $a + \nu$ elements, where the last ν elements are the random vinegar values.

2.9.1 Random values included in the clear

The random value $r \in \{0,1\}^{\bar{l}}$ can be replaced by a subliminal message $m_s \in \{0,1\}^{\bar{l}}$, where $\bar{l} = 128$ for all parameter settings.

2.9.2 Subliminal channel using secret values

Using the verification algorithm we can recover all S_i's, from the first and the X_i's. Then we can recover the random values $v_1, \ldots, v_\nu$, in each (S_i, X_i), using the secret key, which contains $\mathcal{T}$ and c_T.

In each X_j, for $j = 1, \ldots, k$, the subliminal message $m_{s_i} \in \mathbb{F}_q$ replaces the random values v_i, for $i = 1, \ldots, \nu$. The scheme offer three parameter sets. The number of random elements ν is 16, 20 and 28, respectively. The number k is 2 for all sets and the group size q is 2 for all sets. If the signing algorithm does not terminate, for a chosen subliminal message, we can re-encrypt the subliminal message.

2.10 HiMQ – 3

The HiMQ – 3 is based on multivariate quadratic equations. The scheme use a finite field $\mathbb{F}_q$. The affine transformations and maps are denoted in uppercase plain and vectors in lowercase bold.

Signatures have the form

$$\text{Sign}_{HiMQ-3} = T^{-1}(\mathbf{s}),$$

where $T^{-1} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is an invertible affine or linear map, depending on the parameter set, and $\mathbf{s} = (s_1, \ldots, s_\nu, s_{\nu+1}, \ldots, s_n)$ is a solution of the central map $\mathcal{F}$, where the first ν elements of $\mathbf{s}$ are random.

2.10.1 Subliminal channel using secret values

The random elements of $\mathbf{s}$ can be replaced with a subliminal message. The message can be recovered using the secret map T.

The subliminal message $m_{s_i} \in \mathbb{F}_q$ replaces the first ν random values s_i, in $\mathbf{s}$, for $i = 1, \ldots, \nu$. The authors offer three variation the scheme, HiMQ – 3, HiMQ – 3F and HiMQ – 3P, where each variation has one parameter set. The number of group elements q is 2^8, for all variations. The number of random values ν is 31, 24, and 31, respectively. The variations HiMQ – 3F and HiMQ – 3P has a different central map, but the output of the signature algorithms is the same as HiMQ – 3. If the signing algorithm does not terminate we can re-encrypt the chosen subliminal message.

2.11 LUOV

The LUOV, Lifted Unbalanced Oil and Vinegar, scheme is based on the Unbalanced Oil and Vinegar and is a modification of the Oil and Vinegar scheme by Patarin [42]. The scheme uses the finite field $\mathbb{F}_2$ and the extension $\mathbb{F}_{2^r}$. Vectors are denoted in lowercase bold and matrices in uppercase bold.

Signatures have the form

$$\text{Sign}_{LUOV} = \begin{pmatrix} \mathbf{1}_\nu & -\mathbf{T} \\ \mathbf{0} & \mathbf{1}_m \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{o} \end{pmatrix},$$

where $\mathbf{1}_\nu$ and $\mathbf{1}_m$ are identity matrices, $-\mathbf{T}$ is a $\nu \times m$ binary matrix, $\mathbf{v}$ is a vector of random elements, and $\mathbf{o}$ is a unique solution of the central map $\mathcal{F}$.

2.11.1 Subliminal channel using secret values

Let $\mathbf{A} = \begin{pmatrix} \mathbf{1}_\nu & -\mathbf{T} \\ \mathbf{0} & \mathbf{1}_m \end{pmatrix}$, then $\mathbf{A}$ is an upper triangular binary $(\nu + m) \times (\nu + m)$ matrix and is invertible. Using $\mathbf{A}$ we can recover the vector $\mathbf{v}$ by

$$\mathbf{A}^{-1}\text{Sign}_{LUOV} = \begin{pmatrix} \mathbf{v} \\ \mathbf{o} \end{pmatrix}.$$

The vector $\mathbf{v}$ is generated by squeezing a vinegar-sponge, where the sponge is generated using the message and a private seed. Only the sender, which has the private seed, can verify that the vinegar-sponge, and the random values $\mathbf{v}$, is computed according to the protocol. We can replace the random vector $\mathbf{v} \in \mathbb{F}_{2^r}^\nu$ with a subliminal message $\mathbf{m}_s \in \mathbb{F}_{2^r}^\nu$. The scheme offer six parameter sets. The number of random elements ν is 256, 351, 404, 242, 330, and 399, respectively. The field extension r is 8, 8, 8, 48, 64, and 80, respectively. We can re-encrypt the subliminal message if the signing algorithm rejects the produced signature.

2.12 MQDSS

The MQDSS signature scheme is built from the SSH 5-pass identification scheme [46] using the Fiat-Shamir transform. The scheme use the finite field $\mathbb{F}_q$ for an integer q. Vectors are denoted in lowercase bold and matrices in uppercase bold.

Signatures have the form

$$\text{Sign}_{MQDSS} = (R, \sigma_0, \sigma_1, \sigma_2),$$

where R is a random value, σ_0 is a digest of the commitments of $(\mathbf{r}_0^{(j)}, \mathbf{t}_0^{(j)}, \mathbf{e}_0^{(j)})$ and of $(\mathbf{r}_1^{(j)}, \mathbf{F}(\mathbf{t}_0^{(j)} + \mathbf{r}_1^{(j)}) - \mathbf{F}(\mathbf{t}_0^{(j)}) - \mathbf{F}(\mathbf{r}_1^{(j)}) + \mathbf{e}_0^{(j)})$, σ_1 contains r responses of the form $resp_1^{(j)} = (\alpha^{(j)}\mathbf{r}_0^{(j)} - \mathbf{t}_0^{(j)}, \alpha^{(j)}\mathbf{F}(\mathbf{r}_0^{(j)}) - \mathbf{e}_0^{(j)})$, and σ_2 contains r responses of the form $resp_2^{(j)} = \mathbf{r}_{b^{(j)}}^{(j)}$, for $b^{(j)} \in \{0, 1\}$, and r of the commitments in σ_0. The values $\mathbf{r}_0^{(1)}, \ldots, \mathbf{r}_0^{(r)}, \mathbf{t}_0^{(1)}, \ldots, \mathbf{t}_0^{(r)}$, $\mathbf{e}_0^{(1)}, \ldots, \mathbf{e}_0^{(r)}$ are random elements from $\mathbb{F}_q$, generated using a pseudo random generator $PRG_{\mathbf{rte}}$. The values $\mathbf{r}_1^{(j)} = \mathbf{s} - \mathbf{r}_0^{(j)}$, for $j \in \{1, \ldots, r\}$. The first challenge $ch_1 = (\alpha^{(0)}, \ldots, \alpha^{(r)})$ is computed as $H_1(\mathcal{H}(pk||R||M), \sigma_0)$, for hash functions H_1 and $\mathcal{H}$. The second challenge $ch_2 = (b^{(0)}, \ldots, b^{(r)})$ is computed as $H_2(\mathcal{H}(pk||R||M), \sigma_0, ch_1, \sigma_1)$, for a hash function H_2. The multivariate system $\mathbf{F}$ is generated as $XOF_{\mathbf{F}}(S_{\mathbf{F}})$, for an extendable output function XOF, and $S_{\mathbf{F}}$ is the output of a pseudorandom generator $PRG_{sk}(sk)$. The secret key $\mathbf{s}$ is generated as $PRG_{\mathbf{s}}(S_{\mathbf{s}})$, for a pseudorandom generator $PRG_{\mathbf{s}}$, and $S_{\mathbf{s}}$ is the output of a pseudorandom generator $PRG_{sk}(sk)$.

2.12.1 Random values included in the clear

The random value R is generated using a hash function with the message and the secret key as input. Only the sender, which has the secret key, can verify that the random value R is computed according to the protocol. We can replace the random vector with a subliminal message $m_s \in \{0,1\}^k$. The scheme offer two parameter options. The security parameter k is 256 and 384, respectively.

2.12.2 Subliminal channel using secret values

Using the secret key sk the subliminal receiver can recover $\mathbf{r}_0^{(j)}$, $\mathbf{t}_0^{(j)}$, and $\mathbf{e}_0^{(j)}$, for $j \in \{1,\ldots,r\}$, by doing the following.

Generate the second challenge $ch_2 = (b^{(0)},\ldots,b^{(r)})$ using pk, R and M, and $\mathbf{s}$ using sk. These are used to recover the random values $\mathbf{r}_0^{(j)}$'s from $\sigma_2 = (\mathbf{r}_{b^{(1)}}^{(1)},\ldots,\mathbf{r}_{b^{(r)}}^{(r)})$. If $b^{(j)} = 0$ then $\mathbf{r}_0^{(j)} = \mathbf{r}_{b^{(j)}}^{(j)}$, if $b^{(j)} = 1$ then $\mathbf{r}_0^{(j)} = \mathbf{s} - \mathbf{r}_{b^{(j)}}^{(j)}$, for $j \in \{1,\ldots,r\}$.

Generate the first challenge $ch_1 = (\alpha^{(0)},\ldots,\alpha^{(r)})$ using pk, R and M, and the multivariate system $\mathbf{F}$ using sk. These, together with the $\mathbf{r}_0^{(j)}$'s, are used to recover the $\mathbf{t}_0^{(j)}$'s and $\mathbf{e}_0^{(j)}$'s from $\sigma_1 = \{(\alpha^{(j)}\mathbf{r}_0^{(j)} - \mathbf{t}_0^{(j)}, \alpha^{(j)}\mathbf{F}(\mathbf{r}_0^{(j)}) - \mathbf{e}_0^{(j)})\}_{j\in\{1,\ldots,r\}}$. Use $\alpha^{(j)}$ and $\mathbf{r}_0^{(j)}$ to recover $\mathbf{t}_0^{(j)}$, and use $\alpha^{(j)}$ and $\mathbf{F}(\mathbf{r}_0^{(j)})$ to recover $\mathbf{e}_0^{(j)}$, for $j \in \{1,\ldots,r\}$.

Replace the random values with a subliminal message $\mathbf{m}_s \in \{0,1\}^{3rn\lceil\log_2 q\rceil}$. The submission offer two parameter sets. The integer r is 269 and 403, respectively, the integer n is 48 and 64, respectively, and the field order q is 31 for all sets.

2.13 Picnic

The two building blocks of Picnic is a hash function and a block cipher. The hash function is used in a zero knowledge proof and the block cipher is used to generate the public key. The public key is an encryption of a random value, using the secret key as encryption key, and the signature is a proof of knowledge of the secret key using the message as a nonce. The scheme uses a zero knowledge proof to prevent information about the secret key to leak. The scheme specifically use ZKB++, an optimized version of ZKBoo [27], for the zero-knowledge proofs and the block cipher LowMC [2]. The scheme use bytes, byte arrays, integers, integer vectors, and bits, where everything is denoted in plain.

Signatures have the form

$$(e, b_0,\ldots,b_{T-1}, z_0,\ldots,z_{T-1}),$$

where e is a hash of all values used in the computation and the message, the b_i's are parts of the commitments used in the scheme, and the z_i's are parts of the randomness used.

2.13.1 Random values included in the clear

In total there are T triples of randomness used. Each z_i consists of two of the three random values of a triplet, where the sender does not know which two is selected before the signature is generated as the choice depends on the randomness used. Set two of the three random values to be subliminal messages and, using techniques from secret sharing, the third is chosen such that all three values sums to zero. This makes a two out of three threshold scheme and the receiver can recover all three random values from any two values.

Each triplet in the random value are S bits long and there are T triples, hence the subliminal message has length $m_s \in \{0,1\}^{2ST}$. The scheme proposes three security levels. For each security level the value S is 128, 192, and 256, respectively, and the value T is 219, 329, and 438, respectively.

2.14 pqNTRUsign

The pqNTRUsign signature scheme [31, 32] uses the NTRU lattice: the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^N \pm 1\rangle$, for an integer N. An element $\mathbf{x} \in \mathcal{R}_q$ is denoted in bold.

Signatures have the form

$$\text{Sign}_{pqNTRUsign} = \mathbf{r} + (-1)^b\mathbf{af}.$$

Table 2: pqNTRUsign parameters and subliminal message bounds for high orders digits subliminal channel.

Parameter	Description	Parameter set Uniform–1024	Gaussian–1024
N	dimension	1024	1024
B_s	norm bound	98	500
$\|\mathbf{r}\|_\infty$	randomness bound	16384	—
σ	standard deviation	—	250
b	bit	0	$\{0, 1\}$
$\|\mathbf{m}_s\|_\infty$	message space	16	1
$\|r'_i\|$	masking value	$[98, 901]$	—

The random value $\mathbf{r}$ is either sampled from a uniform or a Gaussian distribution, the bit b is either zero or one, the value $\mathbf{a}$ is used to adjust a signature such that it meets a congruence requirement, and the value $\mathbf{f}$ is part of the secret key.

2.14.1 Subliminal channel using high order digits

For a valid signature we know that $\|\mathbf{af}\|_2 \leq B_s$, for a norm bound B_s, and $\|\mathbf{af}\|_\infty \leq \|\mathbf{af}\|_2 \leq B_s$. We can insert a subliminal message in the high order digits of $\mathbf{r}$ if B_s is small compared to the coefficients of $\mathbf{r}$.

Proof of concept – uniform distribution The pqNTRUsign scheme with uniform distribution has the parameters $b = 0$, $B_s = 98$, and $\|\mathbf{r}\|_\infty \leq 16384$, see Table 2. The coefficients of $\mathbf{af}$ is at most three digits long, which leaves us with two digits for the message space. Set the random value $\mathbf{r} = 1000\mathbf{m}_s + \mathbf{r}'$ and

$$\text{Sign}'_{pqNTRUsign} = 1000\mathbf{m}_s + \mathbf{r}' + \mathbf{af},$$

where the bounds on $\mathbf{m}_s$ and coefficient interval of $\mathbf{r}'$ are in Table 2. Note that each coefficient in the masking randomness $\mathbf{r}'$ needs to have the same sign as the subliminal message, that is, $\text{sgn}(r'_i) = \text{sgn}(\mathbf{m}_{s_i})$ for all $i \in \{1, \ldots, N\}$, for the message recovery to be correct. The subliminal message $\mathbf{m}_s$ is bounded such that $\|\text{Sign}'_{pqNTRUsign}\|_\infty \leq \|\text{Sign}_{pqNTRUsign}\|_\infty$, for a valid signature $\text{Sign}_{pqNTRUsign}$, and the coefficients of the masking randomness are chosen such that $0 \leq \|\mathbf{r}' + \mathbf{af}\|_\infty \leq 999$. To recover the subliminal message, the receiver computes

$$\left\lfloor \frac{\text{Sign}'_{pqNTRUsign}}{1000} \right\rfloor$$

to remove the error term and retrieve the subliminal message $\mathbf{m}_s$. The signature is bounded, the max norm should not be too big, and has to meet a congruence requirement. We can re-encrypt the subliminal messages such that these requirements will be met. Resample $\mathbf{r}'$ if $\|1000\mathbf{m}_s + \mathbf{r}'\|_\infty > \|\mathbf{r}\|_\infty$.

Proof of concept – Gaussian distribution The pqNTRUsign scheme with discrete Gaussian distribution has the parameter setting $p = 2$, $B_s = 500$, and $\mathbf{r}$ is sampled from a discrete Gaussian with standard deviation $\sigma = 250$, see Table 2. It is expected that 95 % of the sampled values from the discrete Gaussian to be in the interval $(-500, 500)$, and it is reasonable that 5 % will be outside the interval. If a sampled value $r_i > 500$ then $r_i + a_i f_i > 0$ since $\|\mathbf{af}\|_\infty \leq 500$. Similarly, if $r_i < -500$ then $r_i + a_i f_i < 0$. In other words, to send the bit $\mathbf{m}_{s_i} = 1$ pick a $r_i > 500$ and to send the bit $\mathbf{m}_{s_i} = 0$ pick a $r_i < -500$.

Five percent of $N = 1024$ is 51.2, hence it is reasonable to send 51 bits in the subliminal message. The placement of these 51 bits has to be specified before the signature is sent, say, index set $\{i_1, i_2, \ldots i_{51}\} \subset \{1, 2, \ldots, 1024\}$. The random value is sampled according to the scheme and permuted such that the values at index $i \in \{i_1, i_2, \ldots i_{51}\}$ are larger than 500 or smaller than -500 to send $\mathbf{m}_{s_i} = 1$ and $\mathbf{m}_{s_i} = 0$, respectively. If there not enough large values we can resample the random vector. The signature is required to be bounded, the max norm should not be too big, and meet a congruence. If the produced signature is does not meet theses requirements sample a new random value which will be modified again.

2.15 pqRSA

Post-Quantum RSA is based on factorization and uses large parameters to make it secure in the post quantum setting. Values are represented as byte strings and the computation is over the integers in little endian form.

Signatures have the form

$$\text{Sign}_{pqRSA} = (R, \overline{X}),$$

where R is a uniform random value and $\overline{X}$ is an encoding of $H(R, M)^d \mod N$, for a RSA private key d, modulus N, and message M.

2.15.1 Random values included in the clear

Replace the 32 byte long random string R with an subliminal message m_s.

2.16 pqsigRM

The pqsigRM is a signature scheme based on punctured Reed-Muller code with random insertion and on CFS [17]. All elements are denoted in plain.

Signatures have the form

$$\text{Sign}_{pqsigRM} = (M, e, i_r),$$

where i_r is a random value generated using AES, M is the message, and $e^T = Q^{-1}e'^T$, where e' is a punctured error vector with weight w and Q is a permutation matrix. The error vector is generated by first computing a syndrome s, which is decoded to find a (nonpunctured) error vector and then puncture it to get e'.

2.16.1 Random values included in the clear

Replace the random value i_r with a subliminal message m_s. The exact size of the value is not mentioned in the submission. Looking at the signature size it seems to be 16 bytes long.

2.17 qTesla

The qTesla signature scheme is a variant of the TESLA signature schemes [1, 3, 6], which is based on the scheme by Bai and Galbraith [5]. The hardness of the scheme is based on the decisional ring learning with error problem. The scheme uses the polynomial rings $\mathcal{R} = \mathbb{Z}[x]/\langle x^n + 1\rangle$ and $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$, for a dimension n and integer q. An element in $x \in \mathcal{R}$ is denoted in plain.

Signatures have the form

$$\text{Sign}_{qTesla} = (c', z),$$

where $z = y + sc$ and c' is a hash value. The value $y \in \mathcal{R}_q$ is a uniformly random value. The value $s \in \mathcal{R}_q$ is a secret key sampled from a Gaussian distribution over $\mathcal{R}_q$ with standard deviation σ. The value $c \in \mathcal{R}_q$ is a polynomial with coefficients in $\{-1, 0, 1\}$. It is required for a valid signature that $\|z\|_\infty \leq B - L_S$, for bound parameters B and L_S. Using the signature and the public values a and t the verifier can compute $w = az - t\text{Enc}(c')$, where the value w should be bounded by $q/2 - L_E$ and the L least significant bits of $ay - ec$ is bounded by $2^L - L_E$ for the signature to be valid.

2.17.1 Subliminal channel using secret values

The receiver can recover the random value y with the secret key $sk = (s, e, seed_y, seed_a)$. The sender replaces the random value y with a subliminal message m_s, that is, let $\text{Sign}_{qTesla} = (c', z)$, where $z = m_s + sc$. The receiver computes $sc = sEnc(c')$ and $m_s = z - sc$ to retrieve the subliminal message. If the signature is not valid we can re-encrypt the subliminal message and try again.

2.17.2 Subliminal channel using high order digits

For a valid signature we know that $\|sc\|_\infty \leq L_S$, where L_S is a norm bound parameter. We can insert a subliminal message in the high order digits of y if L_S is small compared to the coefficients of y.

Table 3: qTesla parameters and subliminal message bounds for high orders digits subliminal channel.

Parameter	Description	Parameter set qTesla–128	qTesla–192	qTesla–256
n	dimension	1024	2048	2048
L_S	norm bound	758	1138	1516
B	randomness bound	$2^{20}-1$	$2^{21}-1$	$2^{22}-1$
$\|m_s\|_\infty$	message space	104	209	419
$\|r'_i\|$	masking value	$[758, 9241]$	$[1138, 8861]$	$[1516, 8483]$

Proof of concept In all three parameter settings sc has at most four digits and y has up to seven digits, see Table 3. Set $y = 10000m_s + r'$ and

$$z' = 10000m_s + r' + sc,$$

where the bounds on m_s and r' are given in Table 3. Note that each coefficient in the masking randomness r' needs to have the same sign as the subliminal message, that is, $\text{sgn}(r'_i) = \text{sgn}(m_{s_i})$ for all $i \in \{1, \ldots, n\}$, for the message recovery to be correct. The subliminal message m_s is bounded such that $\|z'\|_\infty \leq B$, and the coefficients of the masking randomness are chosen such that $0 \leq \|r' + sc\|_\infty \leq 9999$. The receiver computes

$$\left\lfloor \frac{z'}{10000} \right\rfloor$$

to remove the error term and retrieve the subliminal message m_s. If the signature is not valid we can re-encrypt the subliminal message. Resample masking values r' such that $\|10000m_s + r'\|_\infty \leq B - L_S$.

2.18 RaCoSS

The RaCoSS signature scheme uses random codes and is based on CFS [17] and KKS [35]. The scheme use binary vectors and matrices, where vectors are denoted in lowercase plain and a matrices in uppercase plain.

Signatures have the form

$$\text{Sign}_{RaCoSS} = (z, c),$$

where c a bit string with hamming weight w, $z = S^t c + y$, S^t is a secret key, and y is a random value. The vector $y \in \{0,1\}^n$ is sampled from the Bernoulli distribution, where $y_i = 1$ with probability ρ and $y_i = 0$ with probability $\rho - 1$ for all indices i. The value $\rho = 0.057$ and y is a sparse vector.

2.18.1 Subliminal channel using secret values

A subliminal receiver with the secret key S^t can recover the random value by computing $y = z - S^t c$. The scheme proposes one parameter set, where $n = 2400$.

To encode a subliminal message in the sparse vector y we divide it into blocks, where a blocks consisting of only zeroes is sending the bit 0 and a block consisting of at east one 1 is sending the bit 1. It is reasonable to see $2400 \cdot 0.057 \approx 137$ ones in a Bernoulli sampled vector, and, assuming messages consists of an almost equal number of ones and zeroes, we can have a block length of 10 we can send 240 bits of information. Sample each block according to the Bernoulli distribution and reject any sample that does not produce the block we want. Concatenate the blocks to make a vector y' and permute its coefficients to produce y. The permutation could be shared together with the secret key or be produced using a hash function with the secret and a counter as input.

2.19 Rainbow

The Rainbow signature scheme [22] is a generalization of the Oil and Vinegar structure. The scheme use a finite field $\mathbb{F}_q$, where vectors are denoted in lowercase bold. The affine transformations and maps are denoted

in uppercase plain letters and their inverse in denoted with the prefix Inv. Field elements are denoted in lowercase plain letters.

Signatures have the form

$$\text{Sign}_{Rainbow} = (InvT \cdot (\mathbf{y} - c_T), r),$$

where $invT$ is the inverse of the affine map $\mathcal{T}$, $\mathbf{y}$ is a solution under the central map $\mathcal{F}$ and consists of ν_1 random values, c_T is a random masking value and a part of the secret key, and $r \in \{0,1\}^l$ is a random salt.

2.19.1 Random values included in the clear

We can replace the random value $r \in \{0,1\}^l$ with a subliminal message, where l is 128 for all parameter sets.

2.19.2 Subliminal channel using secret values

We can insert a subliminal message $\mathbf{m}_s \in \mathbb{F}_q^{\nu_1}$ in the first ν_1 elements of $\mathbf{y}$ and the receiver can recover it by using the map T and masking value c_T, both contained in the secret key.

The scheme offer nine parameter sets. The number of random values ν_1 is 32, 36, 40, 64, 68, 56, 92, 76, and 84, respectively. The number of group elements q is 2^4, 31, 2^8, 31, 2^8, 2^4, 2^8, 2^4, and 31, respectively. The random elements are used to make a solvable system and if the chosen subliminal message does not produce such a system we can re-encrypt the message.

2.20 RankSign

The RankSign signature scheme is based on the RankSign cryptosystem [25], this submission proposes a variation of the existing cryptosystem by adding a small random error in the signature. The scheme use a finite field $\mathbb{F}_{q^m}$, where q is a power of a prime p and m is a positive integer. Vectors are denoted in bold.

Signatures have the form

$$\text{Sign}_{RankSign} = (\mathbf{e}, seed),$$

where $seed$ is a counter and $\mathbf{e}$ is an error vector that satisfies $\mathbf{e}^T\mathbf{H}_{pub} = G(M, seed)$, for a public parity-check matrix $\mathbf{H}_{pub}$, hash function G, and message M.

2.20.1 Random values included in the clear

The seed is a l bits, which can be replaced by a subliminal message m_s. The submission is unclear of the exact value of l, other than it is an integer input to their signature algorithm. From the implementation it seems that the seed use the data type unsigned char, which is one byte. This implies that $l = 8$ for all parameter sets. If the corresponding syndrome (of the error vector $\mathbf{e}$) is not decodable we can re-encrypt the subliminal message.

2.21 SPHINCS$^+$

SPHINCS$^+$ is based on SPHINCS [8], a stateless hash based signature scheme. The scheme outputs byte strings, where a byte string $\mathbf{b}$ is denoted in bold.

Signatures have the form

$$\text{Sign}_{SPHINCS^+} = (\mathbf{R}, \mathbf{SIG}_{FORS}, \mathbf{SIG_{HT}}),$$

where $\mathbf{R}$ is a random value generated using a pseudo random function, $\mathbf{SIG}_{FORS}$ is a FORS signature and $\mathbf{SIG_{HT}}$ is a hyper tree signature. The FORS signature (an improvement of the few times signature scheme HORST [8]) contains private key values, generated using a pseudorandom function, and their associated authentication paths. The hyper tree signature contains XMSS signatures [13]. A XMSS signature contains WOTS signatures [33] and their associated authentication paths, where the WOTS signature takes as input the messages and the public and secret keys.

2.21.1 Random values included in the clear

The random value $\mathbf{R}$ is constructed using a pseudorandom function with a salt, an optional value, and the message as input. The salt is part of the secret key and the verifier cannot verify that the random value was computed using the pseudorandom function. We can replace $\mathbf{R}$ with a subliminal message $\mathbf{m}_s$.

The private key values in $\mathbf{SIG}_{FORS}$ are generated with a pseudorandom function and can be used as a subliminal channel. The signature contains k random values, where each are n bytes.

The submission offers six parameter set. The security parameter n is 16, 24, or 32 bytes long, respectively, and the number of FORS trees k is 10, 30, 14, 33, 22, 30, respectively.

2.22 SRTPI

The SRTPI signature scheme is based on the Non-symmetric Simultaneous Algebraic Riccati Equations problem, which is showed to be NP hard [44]. The scheme works over a field $\mathbb{F}_q$, where matrices are denoted in plain uppercase.

Signatures have the form

$$\text{Sign}_{TPSig} = (m, X_m),$$

where m is the message and X_m is a matrix, where $X_m = X_0 + C^+ M_m + (I - C^+C)U_0(I - CC^+)$. The matrices X_0 and U_0 are random and part of the secret key. The matrix C is random and part of the public key. The matrix C^+ is the Moor-Penrose pseudoinverse of C. The matrix

$$M_m = \begin{bmatrix} I_{n_1} & M_{1,2} & M_{1,3} \\ 0 & -I_{n_2} & L_{2,3} \\ 0 & 0 & 0_{n_3} \end{bmatrix},$$

where $M_{1,2}$ contains the hashed values of the message, in a permuted order, $M_{1,3} = -M_{1,2}L_{2,3} + L_{1,2}L_{2,3} + L_{1,3}$, and I is the identity matrix. The matrices $L_{1,2}$, $L_{1,3}$, and $L_{2,3}$ are random and part of the secret key.

2.22.1 No channel possible

The signature scheme is deterministic and the only difference between two signatures is the messages, and the adversary will notice if any of the secret random matrices is changed for a signature. Any changes in X_0 or U_0 will be detected by a verifier, as the public matrix Q depends them. It is impossible to recover L, M, or π since we need C and C^+ to be invertible, where both C and C^+ are noninvertible by construction. No (reliable) subliminal channel is possible in the SRTPI signature scheme.

2.23 WalnutDSA

The WalnutDSA scheme is based on the Reversing E-Multiplication problem over a braid group. The braid group B_N is defined by the set of Artin generator $\{b_1, b_2, \ldots, b_{N-1}\}$ and a braid $\beta \in B_N$ has the form $\beta = b_{i_1}^{\epsilon_1} b_{i_2}^{\epsilon_2} \cdots b_{i_k}^{\epsilon_k}$, where $i_j \in \{1, 2, \ldots, N-1\}$ and $\epsilon_j \in \{-1, 1\}$. The scheme uses the colored Burau representation,

$$\Pi_{CB} : B_N \to (GL(N, \mathbb{F}_q(t_1, t_1^{-1}, \ldots, t_N, t_N^{-1})) \times S_N),$$

which represents the braid as a $N \times N$ matrix over the ring of Laurent polynomials with N variables over the field $\mathbb{F}_q$ and a permutation $\sigma \in S_N$ of N letters. E-Multiplication is an operation, denoted by $\star$, between a matrix-permutation pair and the colored Burau representation of a braid

$$\star : (GL(N, \mathbb{F}_q) \times S_N) \times (GL(N, \mathbb{F}_q(t_1, t_1^{-1}, \ldots, t_N, t_N^{-1})) \times S_N) \to GL(N, \mathbb{F}_q) \times S_N.$$

When we E-Multiply a colored Burau representation of a braid with a matrix-permutation pair we remove all information of the braid by inserting a set of chosen T-values in the Laurent polynomials.

A signature is a rewritten braid and a hash of the message

$$\text{Sign}_{WalnutDSA} = (H(m), \mathcal{R}(v_1 \cdot w^{-1} \cdot v \cdot E(H(m)) \cdot w' \cdot v_2),$$

where (w, w') is the secret key, $E(H(m))$ is an braid encoding of the hashed message, v, v_1, and v_2 are random values called *cloaking elements*, and $\mathcal{R} : B_N \to B_N$ is a braid rewriting algorithm.

An element ν is a cloaking element for the matrix-permutation pair (M, σ) if $(M, \sigma) \star \Pi_{CB}(\nu) = (M, \sigma)$. The cloaking element is given by $\nu = wb_i^2 w^{-1}$, where b_i is a generator and the corresponding permutation of $w \in B_N$ satisfies $i \mapsto \sigma^{-1}(a)$ and $i + 1 \mapsto \sigma^{-1}(b)$. The integers a, b are public values.

2.23.1 Rewritten random values

The scheme proposes three possible methods for rewriting braids: (1) Rewrite the given braid to the Birman–Ko–Lee (BKL) Normal Form [9] then shorten the braid using Dehornoy's FullHRed algorithm [19]; (2) Stochastic rewriting, that is, randomly rewrite the braid using lookup tables; and (3) Stochastic rewrite first then shorten the braid with FullHRed.

The BKL algorithm rewrites a braid, and every equivalent braid, to its unique normal form. This makes it hard to insert a subliminal message.

The FullHRed algorithm of Dehornoy rewrites the braid, to a fully reduced braid, by removing handles of the form $b_i^{\epsilon} \cdots b_i^{-\epsilon}$, for an Artin generator b_i and $\epsilon \in \{-1, 1\}$. Given a chosen braid $m_s\beta$ we *suspect* that we could use the FullHRed algorithm backward to change the chosen braid to a cloaking element $v_1 = wb_i^2 w^{-1}$. Assuming this is possible, then

$$m_s\beta' = \mathcal{R}(v_1 \cdot w^{-1} \cdot v \cdot E(H(m)) \cdot w' \cdot v_2),$$

where m_s is our chosen subliminal message, $\beta' \in B_N$, and $m_s\beta'$ is a fully reduced braid. We leave it as an open problem, to prove that it is possible to use Dehornoy's algorithm backwards, since the Walnut DSA signature scheme has not been accepted into NIST's second round.

The stochastic rewriting algorithm partitions the braid and replace two consecutive generators $b_{i_j}^{\epsilon_j} b_{i_{j+1}}^{\epsilon_{j+1}}$ in each partition with a relation found in a lookup table. Nothing is changed if no relation was found. We can alter this algorithm such that our chosen subliminal message inserted into the cloaking element stays unchanged and, possibly, write additional messages into the signature.

The random braid w used to construct the cloaking elements has length L, which is 15 and 30 for each parameter set. For option 2 we can at least use three random braids of combined length $3L$ to insert a subliminal message.

3 Summary

In Table 4 we show the bandwidth of each subliminal channel, using the given parameter sets of the proposed signature schemes.

Table 4: The bandwidth of the subliminal channels found. *Public* channels require public values to recover the subliminal message and *Secret* channels needs secret values. A dash (—) denotes no channel possible, a blank space denotes no channel found, and a plus (+) denotes that both the public and secret channel can be used to send a single subliminal messages. The underlined schemes are accepted to the round 2 of NIST's PQCSP.

Signature scheme	Parameter set	Signature length (bits)	Subliminal bandwidth (bits) Public			Secret	
CRYSTAL–Dilithium	I	11096	4624	(41.67 %)		9726	(87.66 %)
	II	16352	6936	(42.41 %)		14590	(89.22 %)
	III	21608	9247	(42.80 %)		19453	(90.03 %)
	IV	26928	11559	(42.93 %)		24317	(90.30 %)
DME	(3,2,24)	144	16	(11.11 %)			
	(3,2,48)	288	32	(11.11 %)			
DRS	all sets						
DualModeMS	128	256016	704	(0.27 %)			
	192	635320	1728	(0.27 %)			
	256	1192232	8192	(0.69 %)			
Falcon	512	4939	320	(6.48 %)			
	768	7951	320	(4.02 %)			
	1024	9866	320	(3.24 %)			
GeMSS	128	384	48	(12.50 %)			
	192	704	80	(11.36 %)			
	256	832	132	(15.87 %)			
Gravity–SPHINCS	S	101120	6400	(6.33 %)			
	M	231432	8448	(3.65 %)			
	L	281344	7424	(1.64 %)			
Gui	184	360	128	(35.56 %)	+	32	(8.89 %)
	312	504	128	(25.40 %)	+	40	(7.94 %)
	448	664	128	(19.28 %)	+	56	(8.43 %)
HiMQ	3	600				248	(41.33 %)
	3F	536				192	(35.82 %)
	3P	536				248	(46.27 %)
LUOV	8-63-256	2552				2048	(80.25 %)
	8-90-351	3528				2808	(79.59 %)
	8-117-404	4168				3232	(77.54 %)
	48-49-242	13600				11616	(85.41 %)
	64-68-330	24800				21120	(85.16 %)
	80-86-399	37600				31920	(84.89 %)
MQDSS	31-48	263056	256	(0.10 %)	+	193680	(73.63 %)
	31-64	542400	384	(0.07 %)	+	386880	(71.33 %)
Picnic	L1FS	272000	56064	(20.61 %)			
	L1UR	431432	56064	(12.99 %)			
	L3FS	613920	126336	(20.58 %)			
	L3UR	974504	126336	(12.96 %)			
	L5FS	1062592	224256	(21.10 %)			
	L5UR	1675792	224256	(13.38 %)			

Signature scheme	Parameter set	Signature length (bits)	Subliminal bandwidth (bits) Public			Secret	
pqNTRUSig	Uniform	11264	4096	(36.36 %)			
	Gaussian	16384	51	(0.31 %)			
pqRSA	sign/pqrsa15	262400	256	(0.10 %)			
	sign/pqrsa20	8388864	256	(≈ 0 %)			
	sign/pqrsa25	268435712	256	(≈ 0 %)			
	sign/pqrsa30	8589934848	256	(≈ 0 %)			
pqsigRM	5-11	2080	128	(6.15 %)			
	6-12	4128	128	(3.10 %)			
	6-13	8224	128	(1.56 %)			
qTesla	128	21760	6861	(31.53 %)		20479	(94.11 %)
	192	45312	15785	(34.84 %)		43006	(94.91 %)
	256	47360	17840	(37.67 %)		45055	(95.13 %)
RaCoSS		4688				240	(5.12 %)
Rainbow	Ia	512	128	(25.00 %)	+	128	(25.00 %)
	Ib	624	128	(20.51 %)	+	178	(28.58 %)
	Ic	832	128	(15.38 %)	+	320	(38.46 %)
	IIIb	896	128	(14.29 %)	+	317	(35.39 %)
	IIIc	1248	128	(10.26 %)	+	544	(43.59 %)
	IVa	736	128	(17.39 %)	+	224	(30.43 %)
	Vc	1632	128	(7.84 %)	+	736	(45.10 %)
	VIa	944	128	(13.56 %)	+	304	(32.20 %)
	VIb	1176	128	(10.88 %)	+	416	(35.39 %)
RankSign	I	11016	8	(0.07 %)			
	II	12008	8	(0.07 %)			
	III	17288	8	(0.05 %)			
	IV	23432	8	(0.03 %)			
SPHINCS$^+$	128s	64640	1408	(2.18 %)			
	128f	135808	3968	(2.92 %)			
	192s	136512	2880	(2.11 %)			
	192f	285312	6528	(2.29 %)			
	256s	238336	5888	(2.47 %)			
	256f	393728	7936	(2.02 %)			
SRTPI	all sets			—			—
WalnutDSA	128 (Option 1)	5173					
	128 (Option 2)	11332	180	(1.59 %)			
	128 (Option 3)	5135					
	256 (Option 1)	9982					
	256 (Option 2)	21557	360	(1.67 %)			
	256 (Option 3)	9932					

References

[1] Sedat Akleylek, Nina Bindel, Johannes Buchmann, Juliane Krämer, and Giorgia Azzurra Marson. An efficient lattice-based signature scheme with provably secure instantiation. Cryptology ePrint Archive, Report 2016/030, 2016. https://eprint.iacr.org/2016/030.

[2] Martin Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. Cryptology ePrint Archive, Report 2016/687, 2016. https://eprint.iacr.org/2016/687.

[3] Erdem Alkim, Nina Bindel, Johannes Buchmann, Özgür Dagdelen, Edward Eaton, Gus Gutoski, Juliane Krämer, and Filip Pawlega. Revisiting TESLA in the quantum random oracle model. Cryptology ePrint Archive, Report 2015/755, 2015. https://eprint.iacr.org/2015/755.

[4] 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.

[5] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In Josh Benaloh, editor, *Topics in Cryptology – CT-RSA 2014*, pages 28–47, Cham, 2014. Springer International Publishing.

[6] Paulo S. L. M. Barreto, Patrick Longa, Michael Naehrig, Jefferson E. Ricardini, and Gustavo Zanon. Sharper ring-LWE signatures. Cryptology ePrint Archive, Report 2016/1026, 2016. https://eprint.iacr.org/2016/1026.

[7] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures-how to sign with rsa and rabin. In *Proceedings of the 15th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'96, pages 399–416, Berlin, Heidelberg, 1996. Springer-Verlag.

[8] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: Practical stateless hash-based signatures. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 368–397, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[9] J S Birman, K H Ko, and Lee Jae Sik. A new approach to the word and conjugacy problems in the braid groups. *Adv. Math.*, 139(math.GT/9712211):322–353. 31 p, Jul 1998.

[10] 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.

[11] 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.

[12] 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.

[13] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. Xmss - a practical forward secure signature scheme based on minimal security assumptions. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, pages 117–129, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[14] 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.

[15] Mike V. D. 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.

[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.

[17] Nicolas T. Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 157–174, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[18] Cryptographic Technology group at NIST. Post-quantum cryptography. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`, 2016. Accessed: 2019-04-29.

[19] Patrick Dehornoy. A fast method for comparing braids. *Advances in Mathematics*, 125(2):200 – 235, 1997.

[20] 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.

[21] Yvo Desmedt. Simmons' protocol is not free of subliminal channels. In *Proceedings of the 9th IEEE Workshop on Computer Security Foundations*, CSFW '96, pages 170–, Washington, DC, USA, 1996. IEEE Computer Society.

[22] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 164–175, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[23] Jintai Ding and Bo-Yin Yang. Degree of regularity for HFEv and HFEv-. In Philippe Gaborit, editor, *Post-Quantum Cryptography*, pages 52–66, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[24] 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.

[25] Philippe Gaborit, Olivier Ruatta, Julien Schrek, and Gilles Zémor. RankSign : an efficient signature algorithm based on the rank metric. *CoRR*, abs/1606.00629, 2016.

[26] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 197–206, New York, NY, USA, 2008. ACM.

[27] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for boolean circuits. Cryptology ePrint Archive, Report 2016/163, 2016. `https://eprint.iacr.org/2016/163`.

[28] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.

[29] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '97, pages 112–131, London, UK, UK, 1997. Springer-Verlag.

[30] Alexander Hartl, Robert Annessi, and Tanja Zseby. Subliminal channels in high-speed signatures. *JoWUA*, 9:30–53, 2018.

[31] Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, and William Whyte. Transcript secure signatures based on modular lattices. Cryptology ePrint Archive, Report 2014/457, 2014. `https://eprint.iacr.org/2014/457`.

[32] Jeffrey Hoffstein, Jill Pipher, William Whyte, and Zhenfei Zhang. A signature scheme from learning with truncation. Cryptology ePrint Archive, Report 2017/995, 2017. `https://eprint.iacr.org/2017/995`.

[33] Andreas Hülsing. W-ots+ – shorter signatures for hash-based signature schemes. In Amr Youssef, Abderrahmane Nitaj, and Aboul Ella Hassanien, editors, *Progress in Cryptology – AFRICACRYPT 2013*, pages 173–188, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[34] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, STOC '89, pages 12–24, New York, NY, USA, 1989. ACM.

[35] Gregory Kabatianskii, E. Krouk, and Ben Smeets. A digital signature scheme based on random error-correcting codes. In *Proceedings of the 6th IMA International Conference on Cryptography and Coding*, pages 161–167, Berlin, Heidelberg, 1997. Springer-Verlag.

[36] 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.

[37] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '09, pages 598–616, Berlin, Heidelberg, 2009. Springer-Verlag.

[38] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. 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 419–453, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[39] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 218–238, New York, NY, 1990. Springer New York.

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

[41] 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.

[42] Jacques Patarin. The Oil and Vinegar signature scheme. presented at the Dagstuhl Workshop on Cryptography, 1997.

[43] Jacques Patarin, Nicolas Courtois, and Louis Goubin. QUARTZ, 128-bit long digital signatures. In David Naccache, editor, *Topics in Cryptology — CT-RSA 2001*, pages 282–297, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[44] Y. Peretz. On multivariable encryption schemes based on simultaneous algebraic riccati equations over finite fields. *Finite Fields Appl.*, 39(C):1–35, May 2016.

[45] Thomas Plantard, Willy Susilo, and Khin Than Win. A digital signature scheme based on $CVP^{\star}_{\infty}$. *International Workshop on Public Key Cryptography*, pages 288–307, 2008.

[46] Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. Public-key identification schemes based on multivariate quadratic polynomials. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, pages 706–723, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[47] C. P. Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 239–252, New York, NY, 1990. Springer New York.

[48] Gustavus J. Simmons. *The Prisoners' Problem and the Subliminal Channel*, pages 51–67. Springer US, Boston, MA, 1984.

[49] 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.

[50] 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.

[51] Gustavus J. Simmons. An introduction to the mathematics of trust in security protocols. In *Computer Security Foundations Workshop VI*, [1993] Computer Security Foundations Workshop VI, pages 121–127. IEEE Computer Society Press, 1993.

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

[53] Alan Szepieniec, Ward Beullens, and Bart Preneel. MQ signatures for PKI. Cryptology ePrint Archive, Report 2017/327, 2017. `https://eprint.iacr.org/2017/327`.

[54] 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.