A Provably Secure Code-based Concurrent Signature Scheme

Maryam Rajabzadeh Asaar[1], Mahmoud Salmasizadeh[1], and Mohammad Reza Aref[2]

[1] Electronics Research Institute (Center), Sharif University of Technology, Tehran, Iran,
[2] Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran.
asaar@ee.sharif.ir, salmasi@sharif.edu, aref@sharif.edu

Abstract. Concurrent signatures allow two entities to generate two signatures in such a way that both signatures are ambiguous till some information is revealed by one of the parties. This kind of signature is useful in auction protocols and a wide range of scenarios in which involving participants are mutually distrustful. In this paper, to have quantum-attack-resistant concurrent signatures as recommended by National Institute of Standards and Technology (NISTIR 8105), the first concurrent signature scheme based on coding theory is proposed. Then, its security is proved under Goppa Parameterized Bounded Decoding and the Goppa Code Distinguishing assumptions in the random oracle model. We should highlight that our proposal can be a post-quantum candidate for fair exchange of signatures without a trusted third party in an efficient way (without a highly degree of interactions).

Keywords: code-based signatures, concurrent signatures, coding theory, provable security, random oracle model.

1 Introduction

Concurrent signatures allow two entities named as an initial signer and a matching signer to generate two ambiguous signatures which are not binding to their real signers till some information called the keystone is released. The main feature of these kinds of signatures is that either both entities are successful at the end of an exchange or not. The notion of concurrent signatures was introduced by Chen, Kudla and Paterson in 2004 [15], and also a provably secure concrete scheme based on Schnorr ring signature scheme [25] was presented. This primitive is a solution to the problem of fair exchange of digital signatures without employing a trusted third party such as auction protocols or fair tendering of contracts [15]. For example, consider a scenario in which B has a contract and put it to tender, and parties A or C would like to propose their price to win the contract. This procedure can be abused by B since it can show A's proposal (the proposed price signed by A) to C to enable it to give a better proposal. With employing concurrent signatures, A present its proposal (the proposed price signed by concurrent signatures by parties A and B), while it keeps the keystone secret. Party B sends another concurrent signature as payment instruction to A to show that B accepts A's proposal. In this case, there is no advantage for B to show A's proposal to C since A's signature is ambiguous and C cannot be convinced who generates the proposal, A or B. If A is the winner, to complete the payment procedure, A needs to release the keystone and at the same time A is committed to do the contract with its proposed price.

Following the work presented by Chen et al. [15], many subsequent concurrent signature schemes have been proposed [14, 9, 28], where their security is based on hard problems in number theory.

In 1994, Shor gave results to show that quantum computers can break security of cryptographic algorithms based on number theory [27]. Recently, lots of research on quantum computers has been done to show that mathematical problems which are difficult for conventional computers can be solved by quantum mechanical phenomena. Therefore, if large-scale quantum computers are ever built, widely-used public-key cryptographic algorithms such as concurrent signatures will be broken. This event seriously will compromise security of digital communications and interactions on the Internet. Post-quantum cryptographic algorithms are secure versions of previous cryptographic ones against both quantum and classical computers. In 2016, National Institute of Standards and Technology (NIST) in an internal report [7] emphasizes the need of switching to post-quantum cryptography. As a consequence, to make classical concurrent signatures resistant against quantum computer threat, it is necessary to have alternative constructions [18, 24, 5] for this primitive. In this paper, we focus on presenting concurrent signatures from coding theory.

In 1978, McEliece introduced the concept of code-based cryptography, and also presented the first code-based public key encryption scheme from the general decoding problem [18]. The proposed scheme [18] cannot be transformed to a signature scheme since it is not invertible. Niederreiter [20] modified McEliece code-based cryptosystem in 1986 such that it can be used to generate a signature scheme. Courtois, Finiasz and Sendrier [8] proposed the first practical code-based signature scheme called CFS scheme in 2001. They adapt the full domain hash approach of Bellare and Rogaway [3] to Niederreither encryption scheme [20] in a way that a message is concatenated with a counter before hashing to make hash values decodable. Although authors presented some security arguments, it does not support provable security. In 2008, Dallot [10] gave a slight modification to their signature scheme in a way that the counter is replaced with a random value, this new scheme is named modified CFS or Dallot scheme, and proved its security under Goppa Parameterized Bounded Decoding [4] and Goppa Code Distinguishing [26] assumptions in the random oracle model [3]. Following the work presented by Dallot [10], several code-based signature schemes with additional properties such as identity-based [6], one-time signatures [2], ring signatures [31], threshold ring [30, 19, 11], blind signatures [21], signcryption scheme [17] and undeniable signature [1] have been proposed, but there is no code-based scheme for concurrent signatures.

Contribution. A concurrent signature is a useful protocol that allows secure, efficient and fair exchange of signatures for legal contracts without requiring a trusted third party. Since security of existing concurrent signature schemes [14, 9, 28] is based on hard problems in number theory, and also it has been proved that cryptography primitives based on number theory are not resistant against quantum attacks [27], in this paper, a concurrent signature scheme from coding theory is proposed. Then, its security is proved under hard problems in coding theory, Goppa Parametrized Bounded Decoding and the Goppa Code Distinguishing problems, in the random oracle model [3]. To the best of our knowledge, this is the first *provably secure* concurrent signature scheme from *coding theory*. To do so, we apply the paradigm "the signer or the matching signer generates a signature" to the Dallot signature scheme [10] to generate ambiguous signatures for two signers and also we use the encryption scheme presented by Niederreiter [20] as a keystone fix generation algorithm. As a consequence, the proposed construction are ambiguous till the keystone is released by one of the signers.

1.1 Organization of the paper

The rest of this paper is organized as follows. Section 2 presents background and complexity assumptions employed as the signature foundation, the outline of concurrent signature algorithm, its protocol and its security model. Our proposed scheme along with its formal security proof and efficiency analysis are given in Section 3 and 4, respectively. Section 5 presents conclusion.

2 Background

In this section, first the used notations in the paper are introduced, then, we review several fundamental backgrounds employed in this research, including coding theory, complexity assumptions, Dallot signature scheme, concurrent signature algorithms and protocol and its security model.

2.1 Notations

In this subsection, the notations used in the paper are defined.

- $\oplus$: X-OR operation.
- $|y|$: the number of bits of the string y.
- $w_H(y)$: the Hamming weight of a word y or the number of non-zero positions of y.
- y^T: transpose of a vector y.
- $\perp$: an empty string.
- $\theta \leftarrow B(y_1, ...)$: the operation of assigning the output of algorithm B on inputs $y_1, ...$ to θ.
- $y \xleftarrow{\$} Y$: the operation of assigning a uniformly random element of Y to y.

2.2 Coding Theory

Let $\mathbb{F}_2$ be the field with two elements and a binary code $\mathcal{C}(n,k)$ be a linear subspace of dimension k of $\mathbb{F}_2^n$, where k and $n \in \mathbb{N}$. Elements of $\mathbb{F}_2^n$ and $\mathcal{C}$ are named words and codewords, respectively. Code $\mathcal{C}(n,k)$ is presented by a $(n-k)\times n$ binary parity check matrix H such that for a codeword $x \in \mathbb{F}_2^n$ belonged to $\mathcal{C}(n,k)$, we have $Hx^T = 0$ and the syndrome of a word $x \in \mathbb{F}_2^n$ is defined as $s = Hx^T$, where $s \in \mathbb{F}_2^{n-k}$. A syndrome s is said to be t-decodable if there exists a word $x \in \mathbb{F}_2^n$ such that $Hx^T = s$ and $w_H(x) \leq t$, where $t = \frac{n-k}{\log_2^n}$ is the error correcting capability of the code $\mathcal{C}(n,k)$.

Goppa codes are a subclass of alternant codes [16], and widely used in code-based cryptography. Goppa codes $G(n,k)$ of t error correcting capability are of length $n = 2^m$ and dimension $k = n - mt$, where m and $t \in \mathbb{N}$. It is assumed that $\mathcal{DEC}_H$ be the decoding algorithm of Goppa code $G(n,k)$ with the parity check matrix H.

2.3 Complexity assumptions

Hard problems and security assumptions used in the paper are defined as follows [10, 12, 26].

Definition 1. Goppa Parameterized Bounded Decoding (GPBD) problem. Given a random $(n-k)\times n$ binary matrix H and a syndrome $s \in \mathbb{F}_2^{n-k}$, output a word $x \in \mathbb{F}_2^n$ such that $w_H(x) \leq \frac{n-k}{\log_2^n}$ and $Hx^T = s$.

Definition 2. Goppa Parametreized Bounded Decoding (GPBD) assumption. The GPBD problem is (τ,ϵ)-hard if there is no algorithm C which runs in time at most τ and with probability at least ϵ breaks the GPBD problem.

Definition 3. Goppa Code Distinguishing (GD) problem. Given a $(n-k)\times n$ binary parity check matrix H, output a bit $b \in \{0,1\}$ indicating if H is a random binary parity check matrix or a Goppa code random binary parity check matrix.

The advantage of the distinguisher C is defined as follows.

$$\begin{aligned} &Adv_C^{GD}(n,k) = \Pr[1 \leftarrow C(H) \mid H \xleftarrow{\$} G(n,k)] - \\ &\Pr[1 \leftarrow C(H) \mid H \xleftarrow{\$} B(n,k)] \end{aligned} \tag{1}$$

Definition 4. Goppa Code Distinguishing (GD) assumption. The GD problem is (τ,ϵ)-hard if there is no algorithm C which runs in time at most τ breaks the GD problem with probability $Adv_C^{GD}(n,k) \geq \epsilon$.

2.4 Dallot signature scheme

In this subsection, we review the modified CFS signature proposed by Dallot [10], Dallot scheme, whose security is based on the GD and GPBD assumptions in the random oracle model [3].

1. Setup: The system parameters are as follows. Let n, k, m and $t \in \mathbb{N}$ be parameters for a Goppa code of length $n = 2^m$, dimension k and error correcting capability $t = \frac{n-k}{\log_2^n}$ such that t-decoding has complexity at least 2^λ for a security parameter λ. Let $g : \{0,1\}^* \to \{0,1\}^{n-k}$ be a random oracle.
   It is assumed that $\tilde{H}$ be a $(n-k)\times n$ parity check matrix of a random binary Goppa code and $\mathcal{DEC}_{\tilde{H}}$ be its t-decoding algorithm. The public key is $pk = H = U\tilde{H}P$, and the secret key is $sk = (\mathcal{DEC}_{\tilde{H}}, U, P)$, where U is a random binary non-singular $(n-k)\times(n-k)$ matrix and P is a random $n \times n$ binary permutation matrix. Therefore, public parameters are $Para = \{n,k,m,t,g\}$.

2. Sign: To create a signature θ on the message $M \in \{0,1\}^*$, the signer picks a number r randomly chosen from $\{1,...,2^{n-k}\}$, computes $\beta = g(r,M)$ and $x = \mathcal{DEC}_{\tilde{H}}(U^{-1}\beta)P$. If $x = \perp$, it chooses another r, and repeats the signing procedure. The signature θ on the message M is (r,x,M).

3. Ver: Given H, $Para$ and a signature $\theta = (r, x, M)$, if $Hx^T = g(r, M)$ and $w_H(x) \leq t$, the signature θ on the message M is valid and outputs 1; otherwise, it outputs 0 and the signature is invalid.

Correctness. The correctness of the signature $\theta = (r, x, M)$ is verified as follows:

$$\begin{aligned} & Hx^T \\ &= (U\tilde{H}P)(\mathcal{DEC}_{\tilde{H}}(U^{-1}\beta)P)^T \\ &= (U\tilde{H}P)P^T(\mathcal{DEC}_{\tilde{H}}U^{-1}\beta)^T \\ &= UU^{-1}\beta \\ &= g(r, M). \end{aligned} \tag{2}$$

2.5 Concurrent signature algorithms

A concurrent signature scheme consists of Setup, KGen, ASign, AVer and Ver algorithms as follows.

- Setup: Given a system security parameter λ, it outputs the set of users $\mathcal{U}$, the message space $\mathcal{M}$, the signature space $\mathcal{R}$, the keystone space $\mathcal{K}$, the keystone fix space $\mathcal{X}$, a function $KGen : \mathcal{K} \to \mathcal{X}$ and other public parameters, π. It also outputs users' public keys pk and each user has its secret key sk; i.e. $(Para, (sk, pk)) \leftarrow Setup(\lambda)$, where $Para = \{\mathcal{U}, \mathcal{M}, \mathcal{R}, \mathcal{K}, \mathcal{X}, \pi, KGen\}$.

- KGen: Given the system's parameter $Para$, a random keystone $\kappa \in \mathcal{K}$, it returns the keystone fix $x \in \mathcal{X}$ such that $x = KGen(\kappa)$.

- ASign: Given the system's parameter $Para$, signer's secret key sk_i and its corresponding public key pk_i, designated user's public key pk_j, a keystone fix $x \in \mathcal{X}$ and the message $M \in \mathcal{M}$ or equivalently an input tuple $(Para, sk_i, pk_i, pk_j, x, M)$, it outputs an ambiguous signature $\theta = (r, x, y)$, where $r \in \mathcal{R}$, $x, y \in \mathcal{X}$; i.e. $\theta \leftarrow ASign(Para, sk_i, pk_i, pk_j, x, M)$.

- AVer: Given the system's parameter $Para$, users' public keys pk_i and pk_j, the signature $\theta = (r, x, y)$ and the message M, returns 1 if θ is valid; otherwise, it returns 0; i.e. $\{0, 1\} \leftarrow AVer(Para, pk_i, pk_j, \theta, M)$.

- Ver: Given the system's parameter $Para$, users' public keys pk_i and pk_j, the signature $\theta = (r, x, y)$, the keystone $\kappa \in \mathcal{K}$ and the message M, and returns 1 if $1 \leftarrow AVer(Para, pk_i, pk_j, \theta, M)$ and also $x = KGen(\kappa)$ holds; otherwise, it returns 0; i.e. $\{0, 1\} \leftarrow Ver(Para, pk_i, pk_j, \theta, \kappa, M)$.

2.6 Concurrent signature protocol

In a concurrent signature protocol, there are two participants A and B. The participant, here A, who generates the keystone and the first ambiguous signature is called the initial signer and the one, here B, who answers to the initial signer by generating another ambiguous signature is called the matching signer. The protocol works as follows. The Setup algorithm is run by A and B to specify system's public parameters, A's public key pk_A and its corresponding secret key sk_A and B's public key pk_B and its corresponding secret key sk_B.

- The initial signer A selects a random keystone $\kappa \in \mathcal{K}$, computes the keystone fix $x = KGen(\kappa)$, and then runs ASign algorithm for the input tuple $(Para, sk_A, pk_A, pk_B, x, M_A)$ to obtain an ambiguous signature $\theta_A = (r_A, x, y_A)$, where $r_A \in \mathcal{R}$, $x, y_A \in \mathcal{X}$. Then, A sends θ_A to B.

- The matching signer B checks if A's ambiguous signature θ_A is valid or not using AVer algorithm. If it is valid, B runs ASign algorithm for the input tuple $(Para, sk_B, pk_B, pk_A, x, M_B)$ to obtain an ambiguous signature $\theta_B = (r_B, x, y_B)$, where $r_B \in \mathcal{R}$, $x, y_B \in \mathcal{X}$. Then, B sends θ_B to A. Note that B uses the same keystone fix $x \in \mathcal{X}$ in its signature.

- The initial signer checks that if B's ambiguous signature θ_B is valid or not using AVer algorithm. If not, A aborts; otherwise, A sends the keystone $\kappa \in \mathcal{K}$ to B. Note that Ver algorithm outputs valid on inputs (θ_A, κ) and (θ_B, κ).

2.7 Security model of concurrent signature schemes

A concurrent signature scheme should be existentially unforgeable under an adaptive-chosen-message attack in the multi-user setting, ambiguous and fair [23].

Unforgeability. To give a formal definition for unforgeability of concurrent signature schemes, the following game between an adversary A and a challenger C is considered to be played [23].

1. Setup: Algorithm C runs the Setup algorithm with a security parameter λ to obtain system's parameter $Para$ and user's key pair (pk, sk), then it sends $(pk, Para)$ to A.

2. The adversary A in addition to making queries to random oracles adaptively issues a polynomially bounded number of questions to the Private Key Extract, KGen, KReveal and ASign oracles as follows.

   - Private Key Extract: Adversary A can ask for the secret key of each user with public key pk, and C in its response returns its corresponding secret key, sk.

   - KGen: Adversary A can ask C to choose a keystone $\kappa \in \mathcal{K}$, and returns its corresponding keystone fix $x = KGen(\kappa)$ to A. Note that A can choose the keystone κ by itself and generate the corresponding keystone fix x using KGen algorithm.

   - KReveal: Adversary A can ask for the keystone $\kappa \in \mathcal{K}$ of each keystone fix $x \in \mathcal{X}$ which was returned by KGen. In response, C returns κ if x was a previous KGen output; otherwise, it returns invalid.

   - ASign: Adversary A can ask for an ambiguous signature on the tuple (pk_i, pk_j, x, M), where $M \in \mathcal{M}$ is the message, $x \in \mathcal{X}$ is the keystone fix and pk_i and pk_j are users' public keys. Then, C returns $\theta = (r, x, y) \leftarrow ASign(Para, sk_i, pk_i, pk_j, x, M)$, where $x, y \in \mathcal{X}$ and $r \in \mathcal{R}$.

     Note that adversary A can generate concurrent signatures in form of (θ, κ) using KGen, ASign and KReveal algorithms for messages and any pairs of users.

3. Eventually, A returns an ambiguous signature $\theta^* = (r^*, x^*, y^*)$ on the message M^* with respect to public keys pk_i^* and pk_j^* such that $1 \leftarrow AVer(Para, pk_i^*, pk_j^*, \theta^*, M^*)$, and wins the forgery game if one of the two following conditions hold:

   Condition 1. Adversary A has not made ASign query for input of $(pk_i^*, pk_j^*, x^*, M^*)$ and $(pk_j^*, pk_i^*, x^*, M^*)$, and also it has not made Private Key Extract query on pk_i^* and pk_j^*.

   Condition 2. Adversary A has not made ASign query on input (pk_i^*, pk_j, x^*, M^*) and Private Key Extract query on input pk_i^*, and the keystone fix x^* was generated by A or KGen oracle.

The formal definition of existential unforgeability of concurrent signatures is given in Definition 5.

Definition 5. A concurrent signature scheme is $(\tau, q_{ro}, q_{pk}, q_{kr}, q_s, \epsilon)$-existentially unforgeable against adaptive chosen message attack in the multi-user setting if there is no adversary which runs in time at most τ, makes at most q_{ro} random oracle queries, q_{pk} Private Key Extract queries, q_{kr} KRveal queries, q_s ASign queries, and can win the forgery game with probability at least ϵ.

Condition 1 of the winning of the adversary in the forgery game models the forgery of an ambiguous signature in a way that it does not know both initial and matching signer's secret keys, and unforgeability in this case ensures that nobody other than those can generate ambiguous signatures. Condition 2 models the forgery in which the adversary knows secret key of one of the signers and tries to cheat the other, and unforgeability in this case guarantees an initial signer or a matching signer cannot forge concurrent signatures on behalf of each other.

Remark 1. To prove unforgeability of the scheme, it is enough to show that it is unforgeable against Condition 2, since the adversary A in Condition 2 is more powerful than the adversary in Condition 1 since it has the secret key of one of the two signers (here sk_j^*) in the concurrent signature protocol in addition to capabilities of the adversary in Condition 1.

Ambiguity. To give a formal definition for ambiguity of concurrent signature schemes, the following game between an adversary A and a challenger C is considered to be played [23].

1. Setup: This is the same as the Setup in the unforgeability game.

2. The adversary A in addition to making queries to random oracles adaptively issues a polynomially bounded number of questions to the Private Key Extract, KGen, KReveal and ASign oracles as explained in the unforgeability game.

3. Adversary A asks for an ambiguous signature of the tuple (pk_i, pk_j, x, M). In response, C chooses $b \in \{i, j\}$ at random, and returns either $\theta_i = (r_i, x, y_i) \leftarrow ASign(Para, sk_i, pk_i, pk_j, x, M)$ or $\theta_j = (r_j, x, y_j) \leftarrow ASign(Para, sk_j, pk_j, pk_i, x, M)$.

4. Eventually, A outputs $b' \in \{i, j\}$, and wins the ambiguity game if $b' = b$ and A has not made a KReveal query on any values of x, y_i or y_j.

The formal definition for ambiguity of concurrent signatures is given in Definition 6.

Definition 6. A concurrent signature scheme is ambiguous if there is no polynomially bounded adversary which can win the ambiguity game with probability non-negligibly greater than $\frac{1}{2}$.

Fairness. To give a formal definition for fairness of concurrent signature schemes, the following game between an adversary A and a challenger C is considered to be played [23].

1. Setup: This is the same as the Setup in unforgeability game.

2. The adversary A in addition to making quires to random oracles adaptively issues a polynomially bounded number of questions to the Private Key Extract, KGen, KReveal and ASign oracles as explained in the unforgeability game.

3. Eventually, A returns a keystone $\kappa^* \in \mathcal{K}$ along with the ambiguous signature $\theta^* = (r^*, x^*, y^*)$ on the message M^* with respect to public keys pk_i^* and pk_j^* such that $1 \leftarrow AVer(Para, pk_i^*, pk_j^*, \theta^*, M^*)$, and wins the fairness game if one of the two following conditions holds:

   Condition 1. Adversary A has not made KRveal query on input x^* which was returned by KGen algorithm, and also $1 \leftarrow Ver(Para, pk_i^*, pk_j^*, \theta^*, \kappa^*, M^*)$.

   Condition 2. Adversary A generates another ambiguous signature $\theta' = (r', x^*, y')$ on the message M' for the public keys pk_i^* and pk_j^* such that $1 \leftarrow AVer(Para, pk_i^*, pk_j^*, \theta', M')$, and also for (κ^*, θ^*), $1 \leftarrow Ver(Para, pk_i^*, pk_j^*, \theta^*, \kappa^*, M^*)$, but $0 \leftarrow Ver(Para, pk_i^*, pk_j^*, \theta', \kappa^*, M')$.

The formal definition for fairness of concurrent signatures is given in Definition 7.

Definition 7. A concurrent signature scheme is fair if the success probability of a polynomially bounded adversary in the fairness game is negligible.

Note that the definition of fairness under Condition 1 guarantees that only the participant who creates the keystone can reveal it to generate a binding signature, and fairness under Condition 2 ensures that each valid ambiguous signature generated using the same keystone fix will become binding by revealing the keystone.

3 Our code-based concurrent signature scheme

In this section, first details of our proposed concurrent signature scheme is presented; then, its security is proved under GPBD and GD assumptions in the random oracle model [3].

3.1 Details of the proposed concurrent signature scheme

In this subsection, we present the details of a concurrent signature scheme. A concurrent signature consists of the following algorithms:

1. Setup: The system parameters are as follows. Let n, k, m and $t \in \mathbb{N}$ be parameters for a Goppa code of length $n = 2^m$, dimension k and error correcting capability $t = \frac{n-k}{\log_2^n}$ such that t-decoding has complexity at least 2^λ for a security parameter λ. Let $g : \{0,1\}^{n-k} \times \{0,1\}^* \to \{0,1\}^{n-k}$ and $h : \{0,1\}^{n-k} \to \{0,1\}^n$ be random oracles, where the latter maps its inputs to vectors such that $w_H(h(.)) \leq t$. It is assumed that $\tilde{H}$ be a $(n-k) \times n$ parity check matrix of a random binary Goppa code and $\mathcal{DEC}_{\tilde{H}}$ be its t-decoding algorithm. The public key is $pk = H = U\tilde{H}P$, and the secret key is $sk = (\mathcal{DEC}_{\tilde{H}}, U, P)$, where U is a random binary non-singular $(n-k)\times(n-k)$ matrix and P is a random $n \times n$ binary permutation matrix. Therefore, public parameters are $Para = \{n, k, m, t, g, h\}$.

2. KGen: Given κ, $w_H(\kappa) \leq t$, as its input, computes $x = h(H_i\kappa^T)$ such that $w_H(x) \leq t$ and returns x.

3. ASign: To generate an ambiguous signature θ on the message $M \in \{0,1\}^*$ and the keystone fix x, the signer i chooses a random number r from $\{1, ..., 2^{n-k}\}$, and computes $\alpha = g(r, M, H_i, H_j) \oplus H_j x^T$ and $y = \mathcal{DEC}_{\tilde{H}_i}(U_i^{-1}\alpha)P_i$. If $y = \perp$, it chooses another r, and repeats the signing procedure. The signature θ on the message M is (r, x, y).

4. AVer: Given $Para$, H_i, H_j and a signature $\theta = (r, x, y)$, the signature θ on the message M is valid and outputs 1 if and only if $H_i y^T \oplus H_j x^T = g(r, M, H_i, H_j)$, $w_H(x) \leq t$, and $w_H(y) \leq t$; otherwise, it outputs 0 and the ambiguous signature is invalid.

5. Ver: Given $\theta = (r, x, y)$ and the keystone κ, the concurrent signature is valid if $h(H_i\kappa^T) = x$, and also $H_i y^T \oplus H_j x^T = g(r, M, H_i, H_j)$, $w_H(x) \leq t$, $w_H(\kappa) \leq t$, and $w_H(y) \leq t$; otherwise, it is invalid.

3.2 Analysis of the proposed scheme

In this subsection, first the correctness of the proposal is verified and then its properties are proved in the random oracle model (see [3] for the background). In order to prove unforgeability of the proposed scheme, we need to show that it is unforgeable against adversary A (as defined in Definition 5).

To prove ambiguity and fairness of our proposed scheme, two lemmas will be given and our main result on the security of the proposed scheme is summarized in Theorem 1.

Correctness. The correctness of the proposed scheme is as follows, and we use $\alpha = g(r, M, H_i, H_j) \oplus H_j x^T$ in what follows.

$$\begin{aligned}
&H_i y^T \oplus H_j x^T \\
&= (U_i \tilde{H}_i P_i)(\mathcal{DEC}_{\tilde{H}_i}(U_i^{-1}\alpha)P_i)^T \oplus H_j x^T \\
&= (U_i \tilde{H}_i P_i) P_i^T (\mathcal{DEC}_{\tilde{H}_i} U_i^{-1}\alpha)^T \oplus H_j x^T \\
&= U_i U_i^{-1}\alpha \oplus H_j x^T \\
&= \alpha \oplus H_j x^T \\
&= g(r, M, H_i, H_j) \oplus H_j x^T \oplus H_j x^T \\
&= g(r, M, H_i, H_j).
\end{aligned} \tag{3}$$

If $\theta = (r, x, y)$ is a valid ambiguous signature on the message M, the relation $H_i y^T \oplus H_j x^T = g(r, M, H_i, H_j)$ holds. If the signature is generated by the jth signer, we can show its correctness in a similar way.

Lemma 1. *If the GPBD problem is $(\tau_{GPBD}, \epsilon_{GPBD})$-hard and GD problem is $(\tau_{GD}, \epsilon_{GD})$-hard, then the proposed scheme is $(\tau, q_h, q_g, q_{pk}, q_{kg}, q_{kr}, \epsilon)$-unforgeable against adversary A such that*

$$\begin{aligned} \epsilon_{GPBD} &\geq \frac{\epsilon - \epsilon_{GD} - q_s(2q_s+q_g)2^{-(n-k)} - 2^{-(n-k)} - (\binom{n}{t})^{-1}}{q_g q_h}, \\ \tau_{GPBD} &\leq \tau + q_s(2mt^2) + q_{kg}(mt^2), \end{aligned} \tag{4}$$

where n, k, t and m are system's constants. In addition, q_g, q_h, q_{pk}, q_{kg}, q_{kr} queries are the number of queries to oracles $g(.)$, $h(.)$ Private Key Extract, KGen, KRveal and ASign, respectively.

Proof. It is assumed that there is an adversary A against unforgeability of the scheme with success probability ϵ. We construct another algorithm C to solve GPBD problem with success probability ϵ_{GPBD}. Given a random binary matrix H^* and a random vector s^*, algorithm C outputs z^* such that $H^*(z^*)^T = s^*$ and $w_H(z^*) \leq t$. Note that substituting the public key of the signer with a random binary matrix H^* changes the success probability of the simulator C with advantage at most ϵ_{BD} to solve the permuted Goppa code distinguishing.

The algorithm C runs Setup on a security parameter λ, and gets a random instance of the GPBD problem, (n, k, m, t, H^*, s^*), to set signer's public key, H_i, to H^* and generate the public parameters $Para = \{n, k, m, t\}$ and invokes the adversary A on $Para$ and $H_i = H^*$. The adversary A runs in time at most τ, makes q_g and q_h queries to the random oracles $g(.)$ and $h(.)$, respectively, and makes q_{pk} queries to the Private Key Extract oracle, q_{kg} queries to the KGen oracle, q_{kr} KRveal queries and q_s queries to the ASign oracle, and can win the unforgeability game with probability at least $\epsilon_1 = \epsilon - \epsilon_{BD}$. Algorithm C maintains initially empty associative tables $T_g[.]$ and $T_h[.]$ to simulate random oracles $g(.)$ and $h(.)$, and answers A's oracle queries as described below.

- $g(.)$ queries: If $T_g[.]$ is defined for query (r, H_i, H_j, M), then, C returns its value; otherwise, C chooses $T_g[r, H_i, H_j, M] \xleftarrow{\$} \{0,1\}^{n-k}$, and returns $g(r, H_i, H_j, M)$ to A.

- $h(.)$ queries: If $T_h[.]$ is defined for the query $H\kappa^T$, then, C returns its value; otherwise, C chooses $T_h[H\kappa^T] \xleftarrow{\$} \{0,1\}^n$ such that its hamming weight is less than or equal to t, $w_H(h(H\kappa^T)) \leq t$, and returns $h(H\kappa^T)$ to A.

- KGen queries: Algorithm C first chooses a random keystone κ such that $w_H(\kappa) \leq t$, computes $H\kappa^T$, makes $h(.)$ query on the input $H\kappa^T$. If $T_h[.]$ has been defined for query $H\kappa^T$, then, C returns its value; otherwise, $T_h[H\kappa^T] \xleftarrow{\$} \{0,1\}^n$, $w_H(h(H\kappa^T)) \leq t$, and returns $x = h(H\kappa^T)$ to A, and maintains the tuple (κ, x) in a list named as K-list.

- KReveal queries: For each keystone fix x generated by a previous KGen query, C looks for the tuple (κ, x) in the K-list, and returns κ; otherwise, C returns invalid.

- Private Key Extract queries: Algorithm C returns the secret key corresponding to the requested public key H if $H \neq H^*$. If $H = H^*$, C outputs invalid.

- ASign queries: For a query (H_i, H_j, x, M), C computes an ambiguous signature following the real ASign algorithm. If $H_i = H^*$, C chooses a random r from $\{1, ..., 2^{n-k}\}$, selects $y \xleftarrow{\$} \{0,1\}^{n-k}$ such that $w_H(y) \leq t$ and computes $\alpha = H^* y^T \oplus H_j x^T$. If $T_g[r, H^*, H_j, M]$ has already been defined, then, C halts, returns $\perp$, and sets $bad \leftarrow true$; otherwise, it sets $T_g[r, H^*, H_j, M] \leftarrow \alpha$, and returns the signature $\theta = (r, x, y)$ on the message M with respect to public keys H^* and H_j to A.

- Finally, A outputs a signature $\theta^* = (r^*, x^*, y^*)$ on the message M^* with respect to public keys H^* and H_j^* with probability ϵ_1, and wins if adversary A has not made ASign query on input (H^*, H_j^*, x^*, M^*), Private Key Extract query on input H^* and the keystone fix x^* was generated by A or by KGen.

The probability of A in returning a forged signature θ^* is $\epsilon_2 = \Pr[E_0]\Pr[E_1|E_0]$ which is computed as follows. First of all, we define events E_0 and E_1.

- E_0 : Algorithm C does not abort as a result of ASign simulation.

– E_1: Adversary A wins the forgery game under Condition 2.

To lower-bound the probability $\Pr[E_0]$ and $\Pr[E_1|E_0]$, we need to compute the probability $\Pr[\neg bad]$, where events *bad* indicate that C aborts in ASign simulation. These probabilities are computed as follows.

Claim 1. $\Pr[E_0] = \Pr[\neg bad] \geq 1 - q_s((q_s + q_g)2^{-(n-k)}) - {q_s}^2 2^{-(n-k)}$.

Proof. The probability of the event E_0 is computed as follows.

– Case 1. If the tuple (r, H^*, H_j, M) generated in one ASign simulation has been occurred by chance in a previous query to the oracle $g(.)$, we have $bad \leftarrow true$. Since there are at most $q_g + q_s$ entries in the table $T_g[.]$ and the number of r, uniformly distributed in $\mathbb{F}_2^{n-k}$, is 2^{n-k}, the probability of this event for one ASign query is at most $(q_g + q_s)2^{-(n-k)}$. Hence, the probability of this event for q_s queries is at most $q_s(q_g + q_s)2^{-(n-k)}$.

– Case 2. If C previously used the same randomness r, uniformly distributed in $\mathbb{F}_2^{n-k}$, in one ASign simulation, we have $bad \leftarrow true$. Since there are at most q_s ASign simulations, this probability is at most $q_s 2^{-(n-k)}$. Therefore, for q_s ASign queries the probability of this event is at most ${q_s}^2 2^{-(n-k)}$.

Claim 3. $\Pr[E_1|E_0] \geq \epsilon_1$.

Proof. The value of $\Pr[E_1|E_0]$ is the probability that A wins the forgery game provided that C does not abort as a result of A's ASign queries. If C did not abort as a result of A's queries, all its responses to those queries are valid. Therefore, by hypothesis A will win the forgery game with probability at least ϵ_1.

Therefore, the probability that A returns a tuple (r^*, x^*, y^*, g) is at least

$$\epsilon_1 - q_s(2q_s + q_g)2^{-(n-k)}.$$

Since g and h are random oracles, the probability of the event that $x^* = h((H^*{\kappa^*}^T))$ is less than $\frac{1}{\binom{n}{t}}$ and also the probability that $g = g(r^*, H^*, H_j^*, M^*)$ is less than $2^{-(n-k)}$, unless they are asked during the attack. Hence, in what follows it is likely that queries (r^*, H^*, H_j^*, M^*) and $(H^*{\kappa^*}^T)$ are asked during a successful attack. The lower bound of probability of wining the forgery game after making queries to g and h oracles is at least

$$\epsilon_1 - q_s(2q_s + q_g)2^{-(n-k)} - 2^{-(n-k)} - (\binom{n}{t})^{-1}.$$

Algorithm C employs A, guesses fixed indices $1 \leq \upsilon_1 \leq q_h$ and $1 \leq \upsilon_2 \leq q_g$, and hopes that υ_1 be the index of the query $(H^*{\kappa^*}^T)$ to oracle h and υ_2 be the index of the query (r^*, H^*, H_j^*, M^*) to the oracle g. Then, C responses with x^* for the query $(H^*{\kappa^*}^T)$ and with $s^* \oplus H_j^* {x^*}^T$ for the query (r^*, H^*, H_j^*, M^*). The probability of these events is $\frac{1}{q_g q_h}$. Since the tuple (r^*, x^*, y^*) is a valid signature, we have $w_H(x^*) \leq t$, $w_H(y^*) \leq t$ and

$$H^*{y^*}^T \oplus H_j^*{x^*}^T = g(r^*, H^*, H_j^*, M^*).$$

With substituting the value of $g(r^*, H^*, H_j^*, M^*) = s^* \oplus H_j^*{x^*}^T$, we have

$$H^*{y^*}^T \oplus H_j^*{x^*}^T = s^* \oplus H_j^*{x^*}^T$$
$$H^*{y^*}^T = s^*$$

with probability at least

$$\frac{\epsilon_1 - q_s(2q_s + q_g)2^{-(n-k)} - 2^{-(n-k)} - (\binom{n}{t})^{-1}}{q_g q_h},$$

where $\epsilon_1 = \epsilon - \epsilon_{GD}$. As a consequence, y^* is a t-decodable of s^*.

Algorithm C's run-time τ_{GPBD} is A's run-time, τ, plus the time required to respond to hash queries, q_{kg} KGen queries and q_s ASign queries. Each KGen simulation takes one syndrome computation and each ASign simulation takes two syndrome computations whose cost is mt^2. Therefore, C's run-time is $\tau_{GPBD} \leq \tau + q_{kg}(mt^2) + q_s(2mt^2)$. This completes the proof.

Lemma 2. *The proposed concurrent signature scheme is ambiguous in the random oracle model.*

Proof. The algorithm C runs Setup on a security parameter λ, and invokes the adversary A on $Para$ and users' public keys. Adversary A issues a polynomially bounded number of random oracle queries, Private Key Extract, KGen, KRveal and ASign queries adaptively as described in the unforgeability game. Then, C chooses two public keys H_i and H_j, and makes ASign query on (H_i, H_j, x, M). In response, C chooses $b \in \{i, j\}$ at random, and returns $\theta_b = (r_b, x, y_b) \leftarrow ASign(Para, sk_b, H_b, H_{\neg b}, x, M)$. Adversary A returns $b' = b$ with probability $\frac{1}{2}$. To show the value of this probability, we compute the probability of ith signer in generating θ_i. Given (H_i, H_j, x, M), the ith signer selects $r_i \xleftarrow{\$} \{1, ..., 2^{n-k}\}$, computes $\alpha = g(r_i, M, H_i, H_j) \oplus H_j x^T$ and $y_i = \mathcal{DEC}_{\tilde{H}_i}(U_i^{-1}\alpha)P_i$, and the signature is (r_i, x, y_i). As a consequence, the probability of the ith signer in creating an ambiguous signatures is $\frac{1}{2^{n-k}}$ and the value of this probability is the same for the jth signer. Therefore, the probability of A in guessing the signature created by the ith signer is indistinguishable from the one generated by the jth signer, and A guesses the real signer of the ambiguous signature with probability $\frac{1}{2}$.

Lemma 3. *The proposed concurrent signature scheme is fair in the random oracle model.*

Proof. The algorithm C runs Setup on a security parameter λ, and invokes the adversary A on $Para$ and users' public keys. The adversary A runs in time at most τ, makes q_g and q_h queries to the random oracles $g(.)$ and $h(.)$, q_{pk} queries to the Private Key Extract oracle, q_{kg} queries to the KGen oracle, q_{kr} KRveal queries and q_s queries to the ASign oracle, and C answers A's oracle queries as described in the unforgeability game. Finally, it is assumed that A wins the fairness game with non-negligible probability, and returns a keystone $\kappa^* \in \mathcal{K}$ along with signature $\theta^* = (r^*, x^*, y^*)$ on the message M^* with respect to public keys H_i^* and H_j^* such that $1 \leftarrow AVer(Para, H_i^*, H_j^*, \theta^*, M^*)$. Since A wins the fairness game, one of the two following conditions holds:

Condition 1. Adversary A has not made KRveal query on input x^* returned by KGen algorithm, and $1 \leftarrow Ver(Para, pk_i^*, pk_j^*, \theta^*, \kappa^*, M^*)$.

Condition 2. Adversary A generates another ambiguous signature $\theta' = (r', x^*, y')$ on the message M' for the public keys pk_i^* and pk_j^* such that $1 \leftarrow AVer(Para, pk_i^*, pk_j^*, \theta', M')$, and also for (κ^*, θ^*), $1 \leftarrow Ver(Para, pk_i^*, pk_j^*, \theta^*, \kappa^*, M^*)$, but $0 \leftarrow Ver(Para, pk_i^*, pk_j^*, \theta', \kappa^*, M')$.

In the following cases, A's output analysis are given based on the aforementioned conditions.

- Case 1 (Analysis of Condition 1). In this case, it is assumed that A with non-negligible probability returns a valid ambiguous signature $\theta^* = (r^*, x^*, y^*)$ along with a keystone κ^* such that for (κ^*, θ^*), $1 \leftarrow Ver(Para, H_i^*, H_j^*, \theta^*, \kappa^*, M^*)$ which this is equivalent to that C has output the tuple (κ^*, x^*) with non-negligible probability such that $x^* = h(H_i^* {\kappa^*}^T)$, where κ^* is a keystone and x^* is the output of KGen algorithm without making KReveal query on x^*. The probability of A in returning the pair (κ^*, x^*) without making KReveal query on x^* is $q_h q_{kg} 2^{-(n-k)}$ and so negligible. This contradicts the assumption that A returns this pair with non-negligible probability.

- Case 2 (Analysis of Condition 2). In this case, it is assumed that A outputs $\theta' = (r', x^*, y')$ as another valid ambiguous signature for public keys H_i^* and H_j^* on the message M' such that for θ',$1 \leftarrow AVer(Para, H_i^*, H_j^*, \theta', M')$, while $0 \leftarrow Ver(Para, H_i^*, H_j^*, \theta', \kappa^*, M')$ on the input (θ', κ^*). Since θ^* is a valid ambiguous signature, AVer for that returns 1, and by assumption Ver algorithm on input (κ^*, θ^*) outputs 1, so $x^* = h(H_i^* {\kappa^*}^T)$ is hold. Because θ^* and θ' shares the same value x^* and also AVer on input θ' returns 1, then $1 \leftarrow Ver(Para, H_i^*, H_j^*, \theta', \kappa^*, M')$ which it is a contradiction with the assumption which states that $0 \leftarrow Ver(Para, H_i^*, H_j^*, \theta', \kappa^*, M')$.

Theorem 1. *The proposed concurrent signature scheme is secure under GPBD and GD assumptions in the random oracle model.*

Proof. The proof follows directly from Lemmas 1, 2 and 3.

4 Efficiency Analysis

Each signer's public key, H, is a $2^m \times mt$ matrix which takes $mt2^m$ bits to be stored, and also the signature θ in our scheme consists of three elements x, y and r, where x and y are $n = 2^m$-bit vectors of weight t which each one takes $\log_2^{\binom{2^m}{t}}$ bits to be stored, and r is a randomly chosen element from $\{1, ..., 2^{mt}\}$ which it takes mt bits to be stored. Hence, the size of the signature θ is $2\log_2^{\binom{2^m}{t}} + mt$. Computational cost of signature scheme is computed as follows. Keystone generation, KGen algorithm, takes one syndrome computation which costs mt^2 bit operations. Ambiguous signature generation, ASign algorithm, requires one syndrome computation and $t!$ decodings which the former costs mt^2 and the latter needs m^3t^2 bit operations. As a consequence, ambiguous signature generation costs $t!(m^3t^2) + mt^2$. Ambiguous signature verification, AVer algorithm, needs two syndrome computation, and so it costs $2mt^2$. Signature verification, Ver algorithm, needs three syndrome computations and so it costs $3mt^2$. In Table 1, KGen, ASign, AVer and Ver computational costs and signature size are summarized.

Table 1. Computation costs of our scheme

Computational Costs	KGen Cost	ASign Cost	AVer Cost	Ver Cost	Signature Size
The proposal	mt^2	$t!m^3t^2 + mt^2$	$2mt^2$	$3mt^2$	$2\log_2^{\binom{2^m}{t}} + mt$

To have an efficient signature scheme, it is recommended that the number of decoding computations for signing messages are reduced, so the parameter t should be small. In 2001, Courtois et al. [8] proposed to use $m = 16$ and $t = 9$, but these parameters are not resistant against the generalized birthday attack [13]. In 2009, Finiasz and Sendrier [13] recommended $m = 22$ and $t = 9$, and with these parameters, the security level is $2^{81.7}$ and the generalized birthday attack is prevented. For parameters $m = 22$ and $t = 9$, each signer's public key is about 99MBytes, signature size will be 557 bits, KGen takes $2^{10.8}$ bit operations, and ASign, AVer and Ver costs $2^{38.19}$, $2^{11.8}$ and $2^{12.38}$ bit operations, respectively. The size of public keys in code-based cryptography (Dallot scheme [10] and our scheme) is large, and can be reduced following the proposals in [29, 22].

5 Conclusion

In this paper, a code-based concurrent signature scheme is proposed, and its security is proved under Goppa Parameterized Bounded Decoding and the Goppa Code Distinguishing assumptions in the random oracle model. We should emphasize that this post-quantum primitive is useful where efficient, secure and fair exchange of signatures without a trusted third party is required, and is widely employed in electronic e-commerce services and auction protocols. However, the size of public keys in our proposal or Dallot scheme [10] is large, many efforts are made to reduce the public key size [29, 22]. As a future work, we focus on presenting other forms of concurrent signature schemes based on coding theory such as its extension to the multi-party case in which several parties can exchange signatures in a fair way concurrently.

References

1. C. Aguilar-Melchor, S. Bettaieb, P. Gaborit, and J. Schrek. A code-based undeniable signature scheme. In *Proc. of the 14th IMA Int. Conf. on Cryptography and Coding-IMACC 2013*, pages 99–119, Oxford, UK, 17-19 December 2013. Springer-Verlag, Berlin.

2. P.S.L.M. Barreto, R. Misoczki, and M. A. Simplicio Jr. One-time signature scheme from syndrome decoding over generic error-correcting codes. *Journal of Systems and Software*, 84(2):198–204, 2011.
3. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. of the 1st ACM Conf. on Computer and Communications Security (CCS 1993)*, pages 62–73, Fairfax, VA, USA, 3-5 November 1993. ACM, New York, NY.
4. E.R. Berlekamp, R.J. McEliece, and H.C.A. Van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.
5. D.J. Bernstein, J. Buchmann, and E. Dahmen. *Post-quantum cryptography*. Springer-Verlag, Berlin, 2009.
6. P. L. Cayrel, P. Gaborit, and M. Girault. Identity-based identification and signature schemes using correcting codes. In *Proc. of the Int. Workshop on Coding and Cryptology (WCC 2007)*, pages 69–78, Versailles, France, 16-20 April 2007. Springer-Verlag, Berlin.
7. L. Chen, S. Jordan, Y.K. Liu, D. Moody, R. Peralta, R.Perlner, and D. Smith-Tone. Report on post-quantum cryptography. Internal Report 8105, National Institute of Standards and Technology, http://dx.doi.org/10.6028/NIST.IR.8105, 2016.
8. N. T. Courtois, M. Finiasz, and N. Sendrier. How to achieve a McEliece-based digital signature scheme. In *Proc. of the 7th Int. Conf. on the Theory and Application of Cryptology and Information Security-Advances in Cryptology-ASIACRYPT 2001*, pages 157–174, Gold Coast, Australia, 9-13 December 2001. Springer-Verlag, Berlin.
9. R. Safavi-Naini D. Tonien, W. Susilo. Multi-party concurrent signatures. In *Proc. of the 9th Int. Conf. on Information Security, ISC 2006*, pages 131–145, Samos Island, Greece, 30 August- 2 September 2006. Springer-Verlag, Berlin.
10. L. Dallot. Towards a concrete security proof of Courtois, Finiasz and Sendrier signature scheme. In *Proc. of the 2nd Western European Workshop on Research in Cryptology-WEWoRC 2007*, pages 65–77, Bochum, Germany, 4-6 July 2008. Springer-Verlag, Berlin.
11. L. Dallot and D. Vergnaud. Provably secure code-based threshold ring signatures. In *Proc. of the 12th Int. Conf. on the Cryptography and Coding*, pages 222–235, Cirencester, UK, 15-17 December 2009. Springer-Verlag, Berlin.
12. M. Finiasz. Nouvelles constructions utilisant des codes correcteurs derreurs en cryptographie clef publique. In *These de doctorat, cole Polytechnique*, Paris, France (in French), October 2004.
13. M. Finiasz and N. Sendrier. Security bounds for the design of code-based cryptosystems. In *Proc. of the 15th Int. Conf. on the Theory and Application of Cryptology and Information Security-Advances in Cryptology-ASIACRYPT 2009*, pages 88–105, Tokyo, Japan, 6-10 December 2009. Springer-Verlag, Berlin.
14. J. Zhou G. Wang, F. Bao. The fairness of perfect concurrent signatures. In *Proc. of 8th Int. Conf. on Information and Communications Security-ICICS 2006*, pages 435–451, Raleigh, NC, USA, 4-7 December 2006. Springer-Verlag, Berlin.
15. K. G. Paterson L. Chen, C. Kudla. Concurrent signatures. In *Proc. of the Int. Conf. on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology-EUROCRYPT 2004*, pages 287–305, Interlaken, Switzerland, 2-6 May 2004. Springer-Verlag, Berlin.
16. F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.
17. K. P. Mathew, S. Vasant, and C. P. Rangan. A provably secure signature and signcryption scheme using the hardness assumption in coding theory. In *Proc. of the 16th Int. Conf. on Information Security and Cryptology-ICISC 2013*, pages 99–119, Seoul, Korea, 27-99 November 2013. Springer-Verlag, Berlin.
18. R.J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN Progress Report 42-44*, (2):114–116, 1978.
19. C.A. Melchor, P.L. Cayrel, P. Gaborit, and F. Laguillaumie. A new efficient threshold ring signature scheme based on coding theory. *IEEE Transactions on Information Theory*, 57(7):4833–4842, 2011.
20. H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986.
21. R. Overbeck. A step towards QC blind signatures. IACR Cryptology ePrint Archive, 2009.
22. P. S. L. M. Barreto R. Misoczki. Compact McEliece keys from Goppa codes. In *Proc. of the 16th Int. Workshop on Selected Areas in Cryptography, SAC 2009*, pages 376–392, Calgary, Canada, 13-14 August 2009. Springer-Verlag, Berlin.
23. H. Wang R. Steinfeld, L. Bull and J. Pieprzyk. Universal designated-verifier signatures. In *Proc. of 9th Int. Conf. on the Theory and Application of Cryptology and Information Security*, pages 523–542, Taipei, Taiwan, 30 November-4 December 2003. Springer-Verlag, Berlin.
24. O. Regev. Lattice-based cryptography. In *Proc. of 26th Annual Int. Cryptology Conf. on Advances in Cryptology-CRYPTO 2006*, pages 131–141, Santa Barbara, California, USA, 20-24 August 2006. Springer-Verlag, Berlin.
25. C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
26. N. Sendrier. Cryptosystmes cl publique bass sur les codes correcteurs derreurs. In *Habilitation diriger les recherches, Universit Pierre et Marie Curie*, Paris, France (in French), March 2002.

27. P.W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proc. of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Santa Fe, New Mexico, USA, 20-22 November 1994. IEEE.
28. W. Susilo, Y. Mu, and F. Zhang. Perfect concurrent signature schemes. In *Proc. of the 6th Int. Conf. on the Information and Communications Security-ICICS 2004*, pages 14–26, Malaga, Spain, 27-29 October 2004. Springer-Verlag, Berlin.
29. P. Gaborit A. Otmani T. P. Berger, P. L. Cayrel. Reducing key length of the McEliece cryptosystem. In *Proc. of the 2nd Int. Conf. on Cryptology in Africa, Progress in Cryptology AFRICACRYPT 2009*, pages 77–97, Gammarth, Tunisia, 21-25 June 2009. Springer-Verlag, Berlin.
30. D.S. Wong, K. Fung, J. K. Liu, and V.K. Wei. On the RS-code construction of ring signature schemes and a threshold setting of RST. In *Proc. of the 5th Int. Conf. on Information and Communications Security- ICICS 2003*, pages 34–36, Huhehaote, China, 10-13 October 2003. Springer-Verlag, Berlin.
31. D. Zheng, X. Li, and K. Chen. Code-based ring signature scheme. *International Journal of Network Security*, 5(2):154–157, 2007.