Computers, Materials & Continua
DOI: 10.32604/cmc.2022.030270
Article

Tech Science Press

Code-based Sequential Aggregate Signature Scheme

Bennian Dou[1,*], Lei Xu[1], Xiaoling Yu[2], Lin Mei[1] and Cong Zuo[3]

[1]School of Mathematics and Statistics, Nanjing University of Science and Technology, Nanjing, 210094, China
[2]College of Data Science, Taiyuan University of Technology, Taiyuan, 030000, China
[3]SCRIPTS, Nanyang Technological University, 639798, Singapore
*Corresponding Author: Bennian Dou. Email: benniandou@163.com
Received: 22 March 2022; Accepted: 29 May 2022

Abstract: This paper proposes the first code-based quantum immune sequential aggregate signature (SAS) scheme and proves the security of the proposed scheme in the random oracle model. Aggregate signature (AS) schemes and sequential aggregate signature schemes allow a group of potential signers to sign different messages respectively, and all the signatures of those users on those messages can be aggregated into a single signature such that the size of the aggregate signature is much smaller than the total size of all individual signatures. Because of the aggregation of many signatures into a single short signature, AS and SAS schemes can reduce bandwidth and save storage; moreover, when a SAS is verified, not only the valid but also the order in which each signer signed can be verified. AS and SAS schemes can be applied to traffic control, banking transaction and military applications. Most of the existing AS and SAS schemes are based either on pairing or Rivest–Shamir–Adleman (RSA), and hence, can be broken by Shor's quantum algorithm for Integer Factoring Problem (IFP) and Discrete Logarithm Problem (DLP). There are no quantum algorithms to solve syndrome decoding problems. Hence, code-based cryptography is seen as one of the promising candidates for post-quantum cryptography. This paper shows how to construct quantum immune sequential aggregate signatures based on coding theory. Specifically, we construct our scheme with the first code based signature scheme proposed by Courtois, Finiasz and Sendrier (CFS). Compared to the CFS signature scheme without aggregation, the proposed sequential aggregate signature scheme can save about 90% storage when the number of signers is asymptotically large.

Keywords: Sequential aggregate signature; CFS signature; post-quantum cryptography

1 Introduction

Security systems may often encounter the scenario which is to manage many different signatures on many different messages generated by many distinct users. For example, in a Public Key

Infrastructure (PKI) of depth n, user signatures are accompanied by a chain of n certificates. The chain contains n signatures by n Certificate Authorities (CAs) on n distinct certificates. Another example is secure border gateway protocol (S-BGP) [1] in which each router receives a list of n signatures attesting to a certain path of length n in the network. A router signs its own segment in the path and passes the resulting list of $n+1$ signatures to the next router. Both systems would benefit from a method which can compress the different signatures on different messages issued by different users. The certificate chain would be shortened by compressing the n signatures in the chain into a short single signature.

Aggregate signatures (AS), proposed by Boneh et al. [2] in 2003, are digital signatures that address the problem of compressing signatures. AS allows n members of a given group of potential signers to sign n different messages m_i respectively, and all the signatures of those users on those messages can be aggregated into a single short signature σ. This single short signature σ and the n original messages m_i are enough to convince the verifier that the n signers did indeed sign the n original messages m_i respectively. In the scheme of [2], the aggregate signature can be produced by anyone. Subsequently, at Eurocrypt 2004, Lysyanskaya et al. [3] proposed another aggregate signature, namely sequential aggregate signature (SAS). In a sequential aggregate signature scheme, the aggregate signature cannot be produced by an outsider; instead, it must be constructed sequentially by each signer modifying the aggregate-so-far signature in turn.

Because of the aggregation of many signatures into a single short signature, AS and SAS schemes can reduce bandwidth and save storage; moreover, when a SAS is verified, not only the valid but also the order in which each signer signed can be verified. AS and SAS schemes can be applied to traffic control, banking transaction and military applications. Particularly, AS and SAS schemes are suitable for the secure border gateway protocol (S-BGP) which is designed to improve the security of the global Internet routing system. In the scenario of S-BGP, each router receives a list of n signatures attesting to a certain path of length n in the network. A router signs its own segment in the path and passes the resulting list of $n+1$ signatures to the next router. When an AS scheme or a SAS scheme is used in S-BGP, one can significantly reduce associated bandwidth overhead and memory space for signatures.

Since Boneh et al. introduced aggregate signature scheme, there are many AS schemes have been presented [2–8], most of them are constructed either from pairings [2,4–8] or from Rivest–Shamir–Adleman (RSA) [3]. The security of the above schemes except [7] is proved in the random oracle model, and the security of [7] is proved in the standard model.

In 1997, Shor [9] discovered a remarkable quantum algorithm which can solve both the factoring problem and the discrete logarithm problem in finite fields and on elliptic curves in polynomial time. So in the age of quantum, Shor's algorithm will break all digital signature schemes, including Full Domain Hash RSA (FDHRSA), the Digital Signature Algorithm (DSA) and the Elliptic Curve DSA (ECDSA), which are based on factoring problem and the discrete logarithm problem. Thus, Shor's algorithm can also break all the aforementioned AS and SAS including Boneh et al.'s AS, Lysyanskaya et al.'s SAS. In response, the cryptographic community has been focusing on a coordinated effort to produce viable alternatives, based on hard problems which are not vulnerable to quantum cryptanalytic attacks. These include cryptosystems based on lattices, linear codes, multivariate equations, and others. Such an effort is led by NIST's (National Institute of Standards and Technology) call for Post-Quantum Standardization [10], which is currently nearing its end.

In1978, McEliece [11] constructed a public key encryption (PKE) based on coding theory. In 1986, Niederreiter [12] proposed an equivalent code-based PKE. But the first practical code-based signature scheme was proposed by Courtois et al. until 2001 [13], their signature scheme is also known as the CFS signature. The CFS signature adapts the full domain hash (FDH) approach of Bellare et al. [14] to

Niederreiter encryption scheme. The security of the CFS signature is based on two difficult problems of coding theory: The *Goppa Bounded Decoding problem* (GBD), and the *Permuted Goppa Code Distinguishing problem* (PGD) [13,15].

Our contribution. Up to date, there exists no quantum algorithm solving the GBD and PGD, so the CFS signature is seen as one of the promising candidates for post-quantum signatures [16,17]. After the publishing of the CFS scheme, there are just a few code-based signatures with special properties have been presented, namely blind, (threshold) ring signature, and identity-based signature, for details, refer to the survey paper [17]. To the best knowledge we know, there is no code-based aggregate signature or sequential aggregate signature in the literature.

In this paper, we construct a code-based sequential aggregate signature from the CFS signature. We prove the proposed SAS secure in the random oracle under reasonable assumptions for permuted Goppa code. Due to the anti-quantum of the CFS signature schemes [16,17], to the best knowledge we know, our scheme is the first code-based quantum immune sequential aggregate signature (SAS) scheme. We compare our scheme with some other aggregate schemes in Tab. 1.

Table 1: The comparisons with other schemes

Scheme	Boneh et al. [2]	Lysyanskaya et al. [3]	Lu et al. [7]	Ours
Type	AS	SAS	SAS	SAS
Hardness assumption	Paring	RSA	Paring	Syndrome decoding
Security model	Random oracle	Random oracle	Standard model	Random oracle
Post-quantum	No	No	No	Yes

Related works. Recently, [18–20] used the Schnorr-Lyubashevsky framework to construct code based signatures without trapdoors. The Schnorr-Lyubashevsky framework has been shown able to produce secure and efficient signature schemes without trapdoors in the lattice-based setting. There are some works [21] analyzed such Schnorr-Lyubashevsky framework code based signatures. Some lattice based aggregate signatures have been proposed in [22,23]. Xu et al. proposed a post-quantum blind signature in [24]. Quantum spin states can be used to construct hybrid cryptosystems and other schemes [25–27].

Organizations. The rest of the paper is organized as follows: In Section 2, we give the formal definitions and security model of sequential aggregate signature schemes. Section 3 reviews coding theory and the CFS signature. A concrete code-based sequential aggregate signature scheme is proposed in Section 4.We compare the efficiency between aggregate and non-aggregate for the CFS signature in Section 5. Finally, we end with concluding remarks in Section 6.

2 Sequential Aggregate Signature Scheme

In this section, we give the formal definitions and security model of sequential aggregate signature (SAS) schemes. We adopt the main notions in [3].

2.1 SAS Schemes

Definition 1. *A sequential aggregate signature (SAS) scheme* **SAS = (KeyGen, AggSign, AggVerify)** *consists of three algorithms*:

KeyGen(1^k)**:** *KeyGen*(1^k) *generates public and secret key pair* (PK_i, SK_i)*of each user* U_i.

AggSign: *AggSign takes as inputs a secret key* SK_i, *a message* $m_i \in \{0,1\}^*$, *and a sequential aggregate-so-far signature* σ *on messages* $m_1, m_2, \cdots, m_{i-1}$ *under respective public keys* $PK_1, PK_2, \cdots, PK_{i-1}$, *where* m_1 *is the inmost message. If* $i = 1$, *the aggregate-so-far* σ *is taken to be empty. It adds a signature on* m_i under SK_i *to the aggregate, outputting a sequential aggregate* σ' *on all messages* $m_1, m_2, \cdots, m_i$.

AggVerify: *AggVerify takes as inputs a sequential aggregate signature* σ, *messages* $m_1, m_2, \cdots, m_i$, *and public keys* $PK_1, PK_2, \cdots, PK_i$, *outputs 1 if* σ *is a valid sequential aggregate (with* m_1 *inmost) on the given messages under the given keys, otherwise outputs 0.*

Fig. 1 shows how a sequential aggregate signature scheme works. SAS schemes can reduce bandwidth and save storage; moreover, when a SAS is verified, not only the valid but also the order in which each signer signed can be verified.

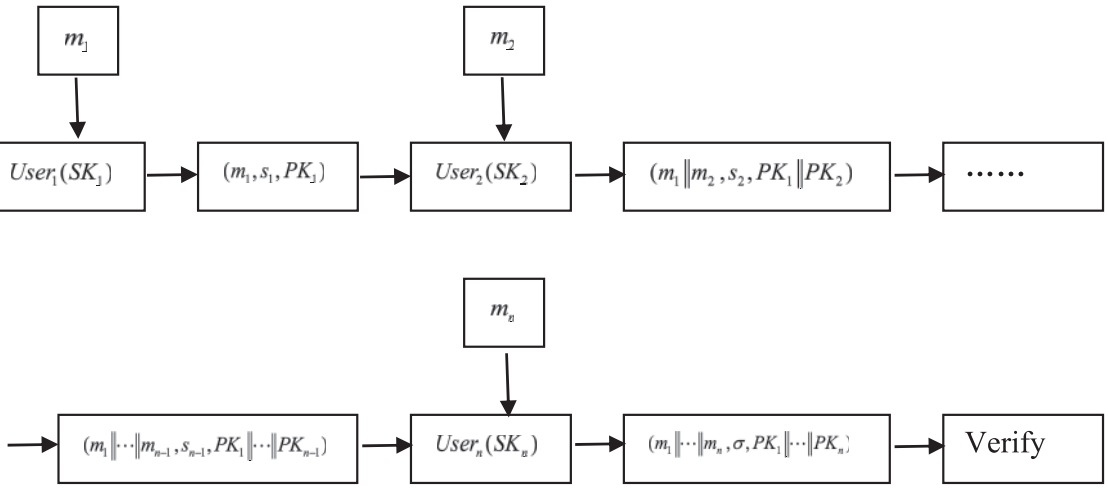

Figure 1: Workflow of a sequential aggregate signature scheme

2.2 Security Model of SAS Schemes

Let **SAS = (KeyGen; AggSign; AggVerify)** be a sequential aggregate signature scheme. To give the security model of SAS schemes, we consider the following game associated to a challenger C and a forger A. A's advantage, $\mathbf{AdvAggSig_A}$, is defined to be his probability to win in the game.

Setup. The aggregate forger A is provided with a public key PK, generated at random.

Queries. Proceeding adaptively, A requests sequential aggregate signatures with PK on messages of his choice. For each query, he supplies a sequential aggregate signature σ on some messages $m_1, m_2, \cdots, m_{i-1}$ under distinct keys $PK_1, PK_2, \cdots, PK_{i-1}$, and an additional message m_i to be signed by the oracle under key PK (where i is at most N, a game parameter).

Response. Finally, A outputs i distinct public keys $PK_1, PK_2, \cdots, PK_i$. Here i is at most N. One of these keys must equal PK, the challenge key. Algorithm A also outputs messages $m_1, m_2, \cdots, m_i$, and a sequential aggregate signature σ by the i users, each on his corresponding message, with PK_1 inmost.

The forger wins if the sequential aggregate signature σ is a valid sequential aggregate signature on messages $m_1, m_2, \cdots, m_i$ under keys $PK_1, PK_2, \cdots, PK_i$, and σ is nontrivial, i.e., A did not request a sequential aggregate signature on messages $m_1, m_2, \cdots, m_{i^*}$ under keys $PK_1, PK_2, \cdots, PK_{i^*}$, where i^* is

the index of the challenge key PK in the forgery. Note that i^* need not equal i. The probability is over the coin tosses of the key-generation algorithm and of A.

Definition 2. *We say a sequential aggregate forger A $(\tau, q_h, q_s, N, \varepsilon)$-breaks an N-user aggregate signature scheme in the sequential aggregate chosen-key model if: A runs in time at most τ; A makes at most q_h queries to the hash function and at most q_s queries to the aggregate signing oracle; $AdvAggSig_A$, is at least ε, and the forged sequential aggregate signature is by at most N users. A sequential aggregate signature scheme is $(\tau, q_h, q_s, N, \varepsilon)$ -secure against existential forgery in the sequential aggregate chosen-key model if no forger $(\tau, q_h, q_s, N, \varepsilon)$-breaks it.*

3 Coding Theory and CFS Signature

3.1 Coding Theory

A binary $[n, k]$-code C is a linear subspace of dimension k of the linear space F_2^n. Elements of F_2^n are called words and elements of C are codewords. A code is usually given in the form of a $(n-k)\times n$ parity check matrix H. The codewords of C are words $\mathbf{x}$ that satisfies $\mathrm{H}\mathbf{x}^T = \mathbf{0}$. A syndrome $\mathbf{s} \in F_2^{n-k}$ is a vector $\mathbf{s} = \mathrm{H}\mathbf{x}^T$ for a word $\mathbf{x}$. The support of a word $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ is defined as $\mathrm{support}(\mathbf{x}) = \{i \,|x_i \neq 0\}$. The Hamming weight of a word $\mathbf{x}$ denoted by $\mathrm{wt}(\mathbf{x})$ is the cardinality of $\mathrm{support}(\mathbf{x})$. The Hamming distance $\mathrm{d}(\mathbf{x}, \mathbf{y})$ between two words $\mathbf{x}, \mathbf{y}$ is defined as $\mathrm{wt}(\mathbf{x}-\mathbf{y})$. The minimum distance d of a $[n, k]$-code C is $\min\{\mathrm{d}(\mathbf{x}, \mathbf{y}) \,|\mathbf{x} \neq \mathbf{y}\}$. A syndrome $\mathbf{s}$ is said to be *decodable* according to a t-error correcting code if there exists a word $\mathbf{x} \in F_2^n$ such that $\mathbf{s} = \mathrm{H}\mathbf{x}^T$ with $\mathrm{wt}(\mathbf{x}) \leq t$. We recall that decoding a syndrome s is retrieving such a word $\mathbf{x}$. Decoding in a random linear code is difficult, which we will restate soon later.

Goppa codes are subfield subcodes of particular alternant codes with efficient decoding algorithm. Goppa codes are widely used in coding-based cryptography. For given integers m and t, Goppa codes are of length $n = 2^m$, of dimension $k = n - mt$ and are t-correcting. The density of decodable syndromes is approximately $\frac{1}{t!}$ [13]. We denote by $Decode_{\mathrm{H}}$ the decoding algorithm associated with a Goppa Code of parity check matrix H.

In order to establish the security of the CFS signature and the pro-posed SAS, we introduce some problems on coding theory.

Definition 3. *Goppa Bouned Decoding Problem (GBD): Given a random $(n-k)\times n$ binary matrix H, a syndrome $\mathbf{s} \in F_2^{n-k}$, output a word $\mathbf{e} \in F_2^n$* of weight $\mathrm{wt(e)} \leq \frac{n-k}{\log_2 n}$ *such that $\mathbf{s} = \mathrm{H}\mathbf{e}^T$.*

The associated decision problem of GBD is NP-complete [28].

Definition 4. *Multiple Goppa Bounded Decoding (MGBD): Given a random $(n-k)\times n$ binary matrix H, an arbitrary number $c > 0$ and a syndrome $\mathbf{s} \in F_2^{n-k}$, output a word $\mathbf{e} \in F_2^n$ of weight* $\mathrm{wt}(\mathbf{e}) \leq c\frac{n-k}{\log_2 n}$ *such that $\mathbf{s} = \mathrm{H}\mathbf{e}^T$.*

The associated decision problem of MGBD is conjectured to be NP-complete, too [29].

Definition 5. *We say an algorithm A is (τ, ε)-solves MGBD(GBD) if A runs in time at most τ, and outputs a word $\mathbf{e} \in F_2^n$ of weight* $\mathrm{wt(e)} \leq c\frac{n-k}{\log_2 n}$ $(\mathrm{wt(e)} \leq \frac{n-k}{\log_2 n})$ *with at least probability ε, such that $\mathbf{s} = \mathrm{H}\mathbf{e}^T$.*

Lemma 1. *There is no algorithm A which is (τ, ε)-solves MGBD and GBD.*

Let $W_{n,t} = \{e \in F_2^n \,|w(e) \leq t\}$, $l = \left\lfloor \log_2 \left| W_{n,t} \right| \right\rfloor$, To describe the CFS signature and the proposed SAS, we need a one to one function $\phi : W_{n,t} \to \{0,1\}^l$. For the construction of such function ϕ and its inverse ϕ^{-1}, refer to [30].

3.2 The CFS Signature

The basic idea of code-based cryptography is using a certain code which has efficient decoding algorithm with trapdoor information as private key and publishing the random look like variant of this code as public key.

The first code-based signature is the CFS signature which was introduced by Courtois, Finiasz and Sendrier at Asiacrypt 2001 [13]. The CFS signature is obtained by applying full hash domain approach to the Niederreiter public key cryptosystem. The CFS signature we will describe below is a minor modification of the original CFS signature, and this modified CFS signature enables us to prove its security more reasonable in random oracle [15,30]. The CFS signature consists of three algorithms which are described as follows:

SystemParameters: Two integers $m, t \in N$.

CFSKeyGen (1^κ): CFSKeyGen (1^κ) generates a $[n, k, d]$-binary Goppa code C ($n = 2^m$, k=n-mt, $d = 2t+1$) with parity check matrix H_0, then C is a t-error correcting code, let its efficient decoding algorithm is $Decode_{\mathrm{H}_0}$. CFSKeyGen (1^κ) generates a random $(n-k) \times (n-k)$ invertible matrix S and a random $n \times n$ permutation matrix P. Let $\mathrm{H} = \mathrm{SH}_0\mathrm{P}$, then H is a parity check matrix of the code C' which is a permuted code of C. C' is also a t-error correcting Goppa code, let its efficient decoding be $Decode_{\mathrm{H}}$(obtained from S, P,$Decode_{\mathrm{H}_0}$). Finally, CFSKeyGen (1^κ) outputs H as the public key, and the corresponding private key is $Decode_{\mathrm{H}}$.

Sign: Let $h : \{0,1\}^* \to \{0,1\}^{n-k}$ be a hash function. Let $r = \lceil \log_2(t!) \rceil$. To sign a message m, Sign does

1. choose $\mathbf{s}$ randomly from $\{0,1\}^r$;
2. $\mathbf{x} = Decode_{\mathrm{H}}(h(m \| \mathbf{s}))$;
3. if no $\mathbf{x}$ was found go to 1.

Let $\mathbf{z} = \phi(\mathbf{x})$ then the signature on m is $(\mathbf{z}, \mathbf{s})$.

Verification: Let $\mathbf{x} = \phi^{-1}(\mathbf{z})$. The signature is valid if and only if $\mathrm{H}\mathbf{x}^T = h(m \| \mathbf{s})$ holds.

Security of the CFS Signature: The security of the CFS signature is based on two assumptions: One is the intractability of GBD as stated in Lemma1; the other is the intractability of the Permuted Goppa Code Distinguish (PGD) which we will describe below.

Definition 6. *Permuted Goppa Code Distinguish Problem (PGD): Given* H *generated by CFSKeyGen (1^κ) as the public key in the CFS signature, the Permuted Goppa Code Distinguish Problem is to distinguish* H *from a parity check matrix of a random linear code.*

Definition 7. *We say an algorithm A is (τ, ε)-solves PGD if A runs in time at most τ, and can distinguish H from a parity check matrix of a random linear code with probability at least $\frac{1}{2} + \varepsilon$.*

Lemma 2 ([13,15]**).** *There is no algorithm A which is (τ, ε)-solves PGD.*

Because of Lemma 2, we can apply Lemma 1 to permuted Goppa code. In fact, we have

Definition 8. *Permuted Goppa Code Bounded Decoding (PGBD): Given H generated by CFSKeyGen (1^κ) as the public key in the CFS signature, an arbitrary number $c > 0$ and a syndrome $\mathbf{s} \in F_2^{n-k}$, the Permuted Goppa Code Bounded Decoding Problem is to output a word $\mathbf{e} \in F_2^n$ of weight* $\mathrm{wt}(\mathbf{e}) \leq c\frac{n-k}{\log_2 n}$(=$ct$, *since n−k=mt, $n = 2^m$*) *such that $\mathbf{s} = \mathrm{H}\mathbf{e}^T$. We say an algorithm A is (τ, ε)-solves PGBD, if A runs in time at most τ, and can output the demanding vector $\mathbf{e}$ with at least probability ε.*

Lemma 3 ([13]**).** *There is no algorithm A which is (τ, ε)-solves PGBD.*

4 The Proposed Code-Based SAS

In this section, we propose a sequential aggregate signature based on CFS signature, and also prove the security of the proposed SAS.

4.1 The Proposed SAS

We firstly introduce some notations for vectors. We write a vector as $\mathbf{x}$ and its length $|\mathbf{x}|$. The elements of $\mathbf{x}$ are $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{|\mathbf{x}|}$. For a vector $\mathbf{x}$, $\mathbf{x}\big|_a^b$ denotes the subvector contains elements $\mathbf{x}_a, \mathbf{x}_{a+1}, \cdots, \mathbf{x}_b$. Suppose $\mathbf{z} \in \{0,1\}^l$, where $l = \left\lfloor \log_2 \left| W_{n,t} \right| \right\rfloor$, if needed, $n-k-l$ 0's can be padded to $\mathbf{z}$ to obtain an $n-k$ dimension vector $\bar{\mathbf{z}} = (\underbrace{00\cdots0}_{n-k-l} \| \mathbf{z})$. Meanwhile, for a vector $\mathbf{z} = (\underbrace{00\cdots0}_{n-k-l} \| \mathbf{z})$, the unpadding of $\mathbf{z}$ is denoted by $\hat{\mathbf{z}} = \mathbf{x}$.Henceforward, we denote by $\mathbf{M}$ the message vector whose elements are $m_1, m_2, \cdots, m_i$, and denote by $\mathbf{H}$ the public key vector whose elements are $\mathrm{H}_1, \mathrm{H}_2, \cdots, \mathrm{H}_i$.

The proposed SAS consists of three algorithms which are described below:

SystemParameters: Two integers $m, t \in N$.

KeyGen (1^κ): For each user U_i $(1 \le i \le n)$, KeyGen (1^κ) generates the key pair $(\mathrm{H}_i, Decode_{\mathrm{H}_i})$ (H_i is the public key, and the corresponding private key is $Decode_{\mathrm{H}_i}$) just as CFSKeyGen (1^κ) does in the CFS signature.

AggSign: The inputs are a secret key $Decode_{\mathrm{H}_i}$, a message $m_i \in \{0,1\}^*$, and a sequential aggregate-so-far signature $\sigma' = (\mathbf{z}', \mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_{i-1})$ on messages $\mathbf{M}$(message vector whose elements are $m_1, m_2, \cdots, m_{i-1}$) under public keys $\mathbf{H}$(public key vector whose elements are H_1,$\mathrm{H}_2, \cdots, \mathrm{H}_{i-1}$). Let $h : \{0,1\}^* \to \{0,1\}^{n-k}$ be a hash function. Let $r = \lceil \log_2(t!) \rceil$. To sign, AggSign does

1. choose $\mathbf{s}_i$ randomly from $\{0,1\}^r$;
2. $\mathbf{x} = Decode_{\mathrm{H}_i}(h(\mathbf{H} \| \mathrm{H}_i, \mathbf{M} \| m_i, \mathbf{s}_i) \oplus \overline{\mathbf{z}'}$;
3. if no $\mathbf{x}$ was found, go to 1.

Let $\mathbf{z} = \phi(\mathbf{x})$, then the sequential aggregate signature on messages $\mathbf{M}$(whose elements are $m_1, m_2, \cdots, m_{i-1}$) under public keys $\mathbf{H}$(whose elements are H_1,$\mathrm{H}_2, \cdots, \mathrm{H}_{i-1}$) is $\sigma = (\mathbf{z}, \mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_i)$.

AggVerify: The input is a sequential aggregate signature $\sigma = (\mathbf{z}, \mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_i)$ on messages $\mathbf{M}$ under public keys $\mathbf{H}$. To verify, let $i = |\mathbf{H}| = |\mathbf{M}|$, set $\mathbf{z}_i = \mathbf{z}$. Then for $j = i, i-1, \cdots, 1$, compute $\mathbf{m}_{j-1} = \mathrm{H}_j(\phi^{-1}(\mathbf{z}_j))^T \oplus (h(\mathbf{H}\big|_1^j, \mathbf{M}\big|_1^j, \mathbf{s}_j)$ and set $\mathbf{z}_{j-1} = \widehat{\mathbf{m}_{j-1}}$. Accept if $\mathbf{z}_0 = \mathbf{0}$.

Correctness of the proposed scheme. From the process of AggVerify, we know that

$$\mathbf{z}_0 = \mathrm{H}_1(\phi^{-1}(\mathbf{z}_1))^T \oplus h(\mathrm{H}_1, m_1, \mathbf{s}_1) \tag{1}$$

On the other hand, if the sequential signature is computed correctly, from the process AggSign we have,

$$\mathbf{z}_1 = \phi(Decode_{\mathrm{H}_1}(h(\mathrm{H}_1, m_1, \mathbf{s}_1) \oplus \mathbf{0}) \tag{2}$$

which implies $\mathrm{H}_1(\phi^{-1}(\mathbf{z}_1))^T \oplus h(\mathrm{H}_1, m_1, \mathbf{s}_1) = \mathbf{0}$. Hence, if the sequential aggregate signature is valid, we have $\mathbf{z}_0 = \mathbf{0}$.

4.2 Security of the Proposed SAS

In this section, we prove the security of the proposed SAS in the random oracle model, we will prove the security of our scheme based on the assumption in lemma 3.

To make the proof easier, we firstly prove the following lemma.

Lemma 4. *There is a forger A* $(\tau, q_h, q_s, N, \varepsilon)$*-breaks the proposed SAS, if and only if there is an forger* $B(\tau', q_h, q_s, N', \varepsilon)$*-breaks the proposed SAS with the target public key is the last key in the key sequence.*

Proof: The lemma can be immediately obtained by the observation that during the AggVerify phase of our scheme, supposing $\sigma = (\mathbf{z}, \mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_i)$ on messages $\mathbf{M}$ under public keys $\mathbf{H}$, each sequential aggregate signature on $\mathbf{M}\big|_1^j$ under $\mathbf{H}\big|_1^j$ $(1 \le j \le i)$ can also be achieved.

Henceforward, we suppose that matrix-vector multiplication, exclusive or of vectors and the computation of ϕ and ϕ^{-1} take unit time. The security of our scheme is obtained from the following theorem.

Theorem 1. *If there is a* $(\tau, q_h, q_s, N, \varepsilon)$ *forger A to break the proposed SAS, then there exists an algorithm* (τ', ε') *B to solve the Permuted Goppa Code Bounded Syndrome Decoding of Problem (PGBD) for* $c = 2$*, with*

$$\varepsilon' \ge (1/2^r)^{N(q_S+1)-q_S}\varepsilon,\ \tau' \le \tau + (q_H + Nq_S + N)(3N+1) + 3N + 2 \tag{3}$$

where $r = \lceil \log_2(t!) \rceil$.

Proof: If A is $(\tau, q_h, q_s, N, \varepsilon)$ forger to break the proposed SAS. From Lemma 4, we can assume, without loss of any generality, that the forgery aggregate signature generated by A has the target public key is the last key i.e., the N-th key in the public key sequence.

Below we will show that if there is an algorithm A is $(\tau, q_h, q_s, N, \varepsilon)$ forger to break the proposed SAS, we can construct an algorithm B which is (τ', ε')-solves PGBD for $c = 2$, which contradicts the assumption in lemma 3. The algorithm B is constructed as follows:

B is given a parity check matrix H of a random [n, k, d]-binary permuted Goppa code ($n = 2^m$, k=n-mt, d=2t+1), a challenge syndrome $\mathbf{y} \in \{0,1\}^{n-k}$, B's goal is to find $\mathbf{x} \in \{0,1\}^n$ with $\mathrm{wt}(\mathbf{x}) \le 2t$ such that $\mathrm{H}\mathbf{x}^T = \mathbf{y}$. B sends H as the target public key to A. B runs A and answers its oracle queries as follows.

Hash queries: The random hash function h has two properties: i.e., the property that the outputs of h are random and the property that one output from $t!$ outputs is decodable. B simulates the hash function as follows.

B maintains a list of tuples $< \mathbf{H}^{(j)}, \mathbf{M}^{(j)}, \mathbf{s}^{(j)}, \mathbf{w}^{(j)}, \mathbf{z}^{(j)}, c_1^{(j)}, c_2^{(j)} >$ as the hash queries' list. We call this list the h-list. When A queries the hash oracle at point $<$ **H**,**M**,**s** $>$ B responds as below:

1. If the query $<$ **H**,**M**,**s** $>$ is already on the h-list, then B finds the entry $< \mathbf{H}, \mathbf{M}, \mathbf{s}, \mathbf{w}, \mathbf{z}, c_1, c_2 >$ and outputs **w** as the answer to the query, i.e., $\mathbf{w} = h(\mathbf{H}, \mathbf{M}, \mathbf{s})$.
2. Otherwise set $i = |\mathbf{H}| = |\mathbf{M}|$, $r = \lceil \log_2(t!) \rceil$, B runs the hashing algorithm on input $< \mathbf{H}\big|_1^{i-1}, \mathbf{M}\big|_1^{i-1} >$ to obtain the corresponding entry $< \mathbf{H}\big|_1^{i-1}, \mathbf{M}\big|_1^{i-1}, \mathbf{s}', \mathbf{w}', \mathbf{z}', c_1', c_2' >$ with $c_1' = 0$ on the h-list. If $i = 1$, B sets $\mathbf{z}' \leftarrow \mathbf{0}$. B must now choose elements along with **H**,**M**,**s** in a new entry on the h-list. Firstly, B chooses a random vector $\mathbf{z}_0 \in \{0,1\}^n$ with weight $\mathrm{wt}(\mathbf{z}_0) \le t$, and B generates a random coin $c_1 \in \{0,1\}$, such that $\Pr[c_1 = 0] = \frac{1}{t!}$. B chooses a random vector $\mathbf{w}_0 \in \{0,1\}^{n-k}$.

2.1 $c_1' = 1$: B sets $\mathbf{w} \leftarrow \mathbf{w}_0$, $\mathbf{z} \leftarrow *$, $c_2 \leftarrow *$ ($*$ is a placeholder value.).

2.2 $c_1' = 0$: If $i < N$ and $c_1 = 1$, B sets $\mathbf{w} \leftarrow \mathbf{w}_0$, $\mathbf{z} \leftarrow *$, $c_2 \leftarrow *$. If $i < N$ and $c_1 = 0$, B sets $\mathbf{w} \leftarrow \mathrm{H}_i\mathbf{z}_0^T \oplus \overline{\phi(\mathbf{z}')}$, and sets $\mathbf{z} \leftarrow \mathbf{z}_0$, $c_2 \leftarrow *$. If $i = N$, B generates another coin $c_2 \in \{0,1\}$ such that

$\Pr[c_2 = 0] = \frac{1}{t!-1}$. If $i = N$, $c_1 = 1$ and $c_2 = 1$, B sets $\mathbf{w} \leftarrow w_0$, $\mathbf{z} \leftarrow *$. If $i = N$, $c_1 = 1$ and $c_2 = 0$, B sets $\mathbf{w} \leftarrow \mathrm{H}\mathbf{z}_0^T \oplus \overline{\phi(\mathbf{z}')} \oplus \mathbf{y}$, $\mathbf{z} \leftarrow \mathbf{z}_0$. If $i = N$, $c_1 = 0$, B sets $\mathbf{w} \leftarrow \mathrm{H}\mathbf{z}_0^T \oplus \overline{\phi(\mathbf{z}')}$, $\mathbf{z} \leftarrow \mathbf{z}_0$ and $c_2 \leftarrow *$.

3. Finally, B adds $< \mathbf{H}, \mathbf{M}, \mathbf{s}, \mathbf{w}, \mathbf{z}, c_1, c_2 >$ to the h-list, and responds to the query as $\mathbf{w} = h(\mathbf{H}, \mathbf{M}, \mathbf{s})$.

In all cases, B's response, $\mathbf{w}$, is indistinguishable from the random oracle.

Aggregate signature queries: A supplies a sequential aggregate signature $\sigma' = (\mathbf{z}', \mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_{N-1})$ on messages $\mathbf{M}\big|_1^{N-1}$ under keys $\mathbf{H}\big|_1^{N-1}$. A requests a sequential aggregate signature on message $\mathbf{M}$ ($\mathbf{M}_N = m$), under keys $\mathbf{H}$ ($\mathbf{H}_N = \mathrm{H}$). Firstly, B uses AggVerify to ensure that σ' is the correct sequential aggregate signature on $\mathbf{M}\big|_1^{N-1}$ under keys $\mathbf{H}\big|_1^{N-1}$. If σ' is incorrect, B responds with a placeholder value $*$. Otherwise, B runs the hash algorithm on ($\mathbf{H}$, $\mathbf{M}$) to obtain $< \mathbf{H}, \mathbf{M}, \mathbf{s}, \mathbf{w}, \mathbf{z}, c_1, c_2 >$. If $c_1 = 0$, B responds to the query with $\sigma \leftarrow (\phi(\mathbf{z}), \mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s})$, we will show later that σ is the correct sequential aggregate signature on ($\mathbf{H}$, $\mathbf{M}$). Otherwise, B reports failure and terminates.

B's output: At last, algorithm A halts and produces a correct forgery sequential aggregate signature $\sigma^* = (\mathbf{z}^*, \mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_N)$ on $(\mathbf{H}^*, \mathbf{M}^*)$, where $\mathbf{H}_N^* = \mathrm{H}$, and $(\mathbf{H}^*, \mathbf{M}^*)$ is not queried to the aggregate signature queries. B now runs the hash algorithm on $(\mathbf{H}^*, \mathbf{M}^*, \mathbf{s}_N)$, and obtains $< \mathbf{H}^*, \mathbf{M}^*, \mathbf{s}, \mathbf{w}, \mathbf{z}, c_1, c_2 >$. If $c_1 = 1$ and $c_2 = 0$, B then computes $\mathbf{x} = \mathbf{z} \oplus \phi^{-1}(\mathbf{z}^*)$ and outputs $\mathbf{x}$, we will show later that $\mathrm{H}\mathbf{x}^T = \mathbf{y}$ with $\mathrm{wt}(\mathbf{x}) \leq 2t$.

This completes the description of algorithm B.

Claim 1. *If A makes a valid sequential aggregate signature query on* ($\mathbf{H}$, $\mathbf{M}$) *with* $\mathbf{H}_N = \mathrm{H}$, *supplying a correct sequential aggregate signature* $\sigma' = (\mathbf{z}', \mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_{N-1})$ *on messages* $\mathbf{M}\big|_1^{N-1}$ *under keys* $\mathbf{H}\big|_1^{N-1}$, *then either B reports failure and halts or B's output* $\sigma \leftarrow (\phi(\mathbf{z}), \mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s})$ *is the correct sequential aggregate signature on* ($\mathbf{H}$, $\mathbf{M}$).

Proof: Firstly, we introduce some notations. For each j, $1 \leq j \leq N$, B's hash algorithm associates with input $(\mathbf{H}\big|_1^j, \mathbf{M}\big|_1^j)$ a tuple $< \mathbf{H}_1^j, \mathbf{M}_1^j, \mathbf{s}^{(j)}, \mathbf{w}^{(j)}, \mathbf{z}^{(j)}, c_1^{(j)}, c_2^{(j)} >$. If B does not abort, then $c_1^{(j)} = 0$, $c_2^{(j)} = *$ for $1 \leq j \leq N$. It is reasonable to assume that $(\mathbf{H}\big|_1^j, \mathbf{M}\big|_1^j)$ is queried to the hash query before it is signed, for the hash function is random oracle. Hence, from the hash algorithm, we obtain that $\mathbf{z} = \mathbf{z}^{(N)}, \mathbf{z}' = \phi(\mathbf{z}^{(N-1)})$, $\mathbf{s} = \mathbf{s}^{(N)}$ and $\mathbf{s}^{(j)} = \mathbf{s}_j$ $(1 \leq j \leq N-1)$. Also from the hash algorithm, we know that

$$\mathbf{z}' = \phi(\mathbf{z}^{(N-1)}) = h(\mathbf{H}_1^{\mathrm{N}}, \mathbf{M}_1^{\mathrm{N}}, \mathbf{s}) \oplus \mathrm{H}\mathbf{z}^T \tag{4}$$

$$\mathbf{z}^{(j)} = \phi^{-1}(h(\mathbf{H}_1^{\mathrm{j+1}}, \mathbf{M}_1^{\mathrm{j+1}}, \mathbf{s}^{(j+1)}) \oplus \mathrm{H}_{j+1}(\mathbf{z}^{(j+1)})^T) \tag{5}$$

$$\mathbf{z}^{(j)} = \phi^{-1}(h(\mathbf{H}_1^{\mathrm{j+1}}, \mathbf{M}_1^{\mathrm{j+1}}, \mathbf{s}^{(j+1)}) \oplus \mathrm{H}_{j+1}(\mathbf{z}^{(j+1)})^T)(1 \leq j \leq N-2) \tag{6}$$

$$\mathrm{H}_1(z^{(1)})^T \oplus h(\mathrm{H}_1, m_1, \mathbf{s}^{(1)}) = \mathbf{0}$$

The last equation shows that $\sigma \leftarrow (\phi(\mathbf{z}), \mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s})$ is a valid sequential aggregate signature on ($\mathbf{H}$, $\mathbf{M}$).

Claim 2: *If A outputs a valid nontrivial forgery sequential aggregate signature* $\sigma^* = (\mathbf{z}^*, \mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_N)$ *on* $(\mathbf{H}^*, \mathbf{M}^*)$, *then B either reports failure and halts or outputs the correct solution* $\mathbf{x}$ *such that* $\mathrm{H}\mathbf{x}^T = \mathbf{y}$ *with* $\mathrm{wt}(\mathbf{x}) \leq 2t$.

Proof: We use the same notations introduced in Claim 1. If B does not abortion, then $c_1^{(N)} = 0$, $c_2^{(N)} = 0$. Hence, from the hash algorithm, we know that

$$h(\mathbf{H}^*, \mathbf{M}^*, \mathbf{s}_N) = \mathrm{H}(\mathbf{z}^{(N)})^T \oplus \overline{\phi(\mathbf{z}^{(N-1)})} \oplus \mathbf{y} \tag{7}$$

Thus,

$$\begin{aligned}\mathrm{H}(\phi^{-1}(\mathbf{z}^*))^T &= h(\mathbf{H}^*, \mathbf{M}^*, \mathbf{s}_N) \oplus \phi(\mathbf{z}^{(N-1)}) \\ &= \mathrm{H}(\mathbf{z}^{(N)})^T \oplus \phi(\mathbf{z}^{(N-1)}) \oplus \mathbf{y} \oplus \phi(\mathbf{z}^{(N-1)}) \\ &= \mathrm{H}(\mathbf{z}^{(N)})^T \oplus \mathbf{y} \end{aligned} \tag{8}$$

And hence, we can obtain that if we set $\mathbf{x} = \mathbf{z}^{(N)} \oplus \phi^{-1}(\mathbf{z}^*)$, then $\mathrm{H}\mathbf{x}^T = \mathbf{y}$ with $\mathrm{wt}(\mathbf{x}) \leq 2t$.

It remains to show that the success probability ε' that B outputs $\mathbf{x}$ is at least $(1/2^r)^{N(q_S+1)-q_S}\varepsilon$. We define three events as below:

E_1: B does not abort during the q_s aggregate signature queries phase.

E_2: A outputs a valid and nontrivial forgery sequential aggregate signature.

E_3: B does not abort during the output phase.

Then B succeeds in giving the correct solution if all of the above events happen, i.e., $\varepsilon' = \Pr[E_1 \wedge E_2 \wedge E_3] = \Pr[E_1] \cdot \Pr[E_2 | E_1] \cdot \Pr[E_3 | E_1 \wedge E_2]$.

On the other hand, we have the following three claims.

Claim 3. $\Pr[E_1] = (\frac{1}{2^r})^{(N-1)q_S}$.

Proof: If B does not abort during each aggregate signature query, then all $c_1^{(j)} = 0 (1 \leq j \leq N-1)$, which implies the probability that B does not abort during each signature query is $(\frac{1}{2^r})^{(N-1)}$. Hence, the probability that B does not abort during the q_s signature queries is $\Pr[E_1] = (\frac{1}{2^r})^{(N-1)q_S}$.

Claim 4. $\Pr[E_2 | E_1] \geq \varepsilon$.

Proof: Trivially.

Claim 5. $\Pr[E_3 | E_1 \wedge E_2] = (\frac{1}{2^r})^N$.

Proof: If B does not abort during the output phase, then all $c_1^{(j)} = 0 (1 \leq j \leq N-1)$, $c_2^{(N)} = 0$ which implies the probability that B does not abort during the output phase is $(\frac{1}{2^r})^{(N-1)} \cdot \frac{1}{2^r} = (\frac{1}{2^r})^N$. Hence, $\Pr[E_3 | E_1 \wedge E_2] = (\frac{1}{2^r})^N$.

Combine the results of the above claims, we obtain immediately that $\varepsilon' \geq (1/2^r)^{N(q_S+1)-q_S}\varepsilon$.

The running ting τ' of B equal to the running time τ plus the time of B's time to respond A's q_H hash queries and q_s aggregate signature queries, each hash query involves at most N matrix-vector multiplication, at most $N+1$ exclusive or of vectors and at most N computation of the function ϕ. Each aggregate signature query needs at most N hash queries, at most N computation of ϕ and ϕ^{-1}, N exclusive or of vectors. At last, in order to give $\mathbf{x}$, B needs one exclusive or of two vectors and N hash queries. Then, the total time of B's is at most

$$\tau + (q_H + Nq_S + N)(3N+1) + 3N + 2 \tag{9}$$

5 Efficiency Comparisons

We now compare the efficiency between our scheme and the original CFS scheme without aggregate in N users setting. To achieve security, for the CFS signature, [31] suggests the parameters: $m = 22$, $t = 9$, $n = 2^{22}$, and for each signature $(\mathbf{z}, \mathbf{s})$, one needs about 182 and 19 bits to store $\mathbf{z}$ and $\mathbf{s}$ respectively. So in order to store N signatures, the total bits needed in the CFS scheme is $(182+19)N = 201N$, while one needs only $182 + 19N$ bits to store the signatures by using the

sequential aggregate signatures we proposed. Hence, one can save about $(1-\frac{19}{201})\approx 90\%$ storage when N is big. It is not difficult to see that the signature generation and verification are of the same time. We give the comparison in Tab. 2.

Table 2: Efficiency comparison ($m=22$, $t=9$, $n=2^{22}$)

	Signature numbers	Signature length
Plain CFS	N	$201\,N$
Aggregate CFS	N	$182+19\,N$

6 Conclusions and Open Problems

In this paper, we propose the first code based sequential aggregate signature, and prove the security of the scheme in the random oracle model. We note that the length of the aggregate signature in our scheme is not constant. It is an open problem to construct code based sequential aggregate signatures with constant length. It is also an interesting problem to construct code based aggregate signatures using the Schnorr-Lyubashevsky Framework [18,20].

Funding Statement: This work was supported in part by the National Natural Science Foundation of China under Grant 62072240, by the Natural Science Foundation of Jiangsu Province under Grant BK20210330 and by the National Key Research and Development Program of China under Grant 2020YFB1804604.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. Kent, C. Lynn and K. Seo, "Secure border gateway protocol (secure-BGP)," *IEEE J. Selected Areas in Comm*, vol. 18, no. 4, pp. 582–92, 2000.

[2] D. Boneh, C. Gentry, B. Lynn and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Advances in Cryptology, EUROCRYPT 2003. Proc.: Lecture Notes in Computer Science (LNCS 2656)*, Warsaw, Poland, pp. 416–432, 2003.

[3] A. Lysyanskaya, S. Micali, L. Reyzin and H. Shacham, "Sequential aggregate signatures from trapdoor permutations," in *Advances in Cryptology, EUROCRYPT 2004. Proc.: Lecture Notes in Computer Science (LNCS 3027)*, Interlaken, Switzerland, pp. 74–90, 2004.

[4] A. Boldyreva, C. Gentry, A. O'Neill and D. Yum, "Ordered multisignatures and identity-based aggregate signatures with applications to secure routing," in *Proc. of the 14th ACM Conf. on Computer and Communications Security, CCS 2007*, Alexandria, Virginia, USA, pp. 276–285, 2007.

[5] X. Cheng, J. Liu and X. Wang, "Identity-based aggregate and verifiably encrypted signatures from bilinear pairing," in *Int. Conf. on Computational Science and Its Applications, ICCSA 2005. Proc.: Lecture Notes in Computer Science (LNCS 3483)*, Singapore, pp. 1046–1054, 2005.

[6] C. Gentry and Z. Ramzan, "Identity-based aggregate signatures," in *Public Key Cryptography, PKC 2006. Proc.: Lecture Notes in Computer Science (LNCS 3958)*, New York City, NY, USA, pp. 257–273, 2006.

[7] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham and B. Waters, "Sequential aggregate signatures and multisignatures without random oracles," in *Advances in Cryptology, EUROCRYPT 2006. Proc.: Lecture Notes in Computer Science (LNCS 4004)*, St. Petersburg, Russia, pp. 465–485, 2006.

[8] J. Xu, Z. Zhang and D. Feng, "ID-Based aggregate signatures from bilinear pairings," in *Cryptology and Network Security, 4th Int. Conf., CANS 2005. Proc.: Lecture Notes in Computer Science (LNCS 3810)*, Xiamen, China, pp. 110–119, 2005.
[9] P. W. Shor, "Polynomial-tme algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Computing*, vol. 26, pp. 1484–1509, 1997.
[10] National Institute of Standards and Technology, "NIST post quantum standardization process," 2017. [Online]. Available: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography.
[11] R. J. McEliece, "A Public-key cryptosystem based on algebraic coding theory," *Jet propulsion Laboratory DSN Progress Report*, vol. 42-44, pp. 114–116, 1978.
[12] H. Niederreiter, "Knapsack-type crytosystems and algebraic coding theory," *Prob. Contr. Inform. Theory*, vol. 15, no. 2, pp. 157–166, 1986.
[13] N. Courtois, M. Finiasz and N. Sendrier, "How to achieve a McEliece-based digital signature scheme," in *Advances in Cryptology, ASIACRYPT 2001. Proc.: Lecture Notes in Computer Science (LNCS 2248)*, Gold Coast, Australia, pp. 157–174, 2001.
[14] 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*, Fairfax, Virginia, USA, pp. 62–73, 1993.
[15] L. Dallot, "Towards a concrete security proof of courtois, finiasz and sendrier signature scheme," in *Research in Cryptology, Second Western European Workshop, WEWoRC 2007. Proc.: Lecture Notes in Computer Science (LNCS 4945)*, Bochum, Germany, pp. 65–77, 2008.
[16] J. Buchmann, C. Coronado, M. Doring, D. Engelbert, C. Ludwig *et al.,* "Post-quantum signatures," 2004. [Online]. Available: https://eprint.iacr.org/2004/297.pdf.
[17] P. Cayrel and M. Meziani, "Post-quantum cryptography: Code-based signatures," in *Advances in Computer Science and Information Technology, AST/UCMA/ISA/ACN 2010. Proc.: Lecture Notes in Computer Science (LNCS 6059)*, Miyazaki, Japan, pp. 82–89, 2010.
[18] Y. Song, X. Huang, Y. Mu, W. Wu and H. Wang, "A Code-based signature scheme from the lyubashevsky framework," *Theoretical Computer Science*, vol. 835, pp. 15–30, 2020.
[19] Z. Li, C. Xing and S. L. Yeo, "A new code based signature scheme without trapdoors," *Cryptology ePrint Archive*, Report 2020/1250, 2020.
[20] M. Baldi, F. Chiaraluce and P. Santini, "Code-based signatures without trapdoors through restricted vectors," *Cryptology ePrint Archive*, Report 2021/294, 2021.
[21] M. Baldi, J. C. Deneuville, E. Persichetti and P. Santini, "Cryptanalysis of a code-based signature scheme based on the schnorr-lyubashevsky framework," *IEEE Communications Letters*, vol. 25, no. 9, pp. 2829–2833, 2021.
[22] R. Bansarkhani and J. Buchmann, "Towards lattice based aggregate signatures," in" *Progress in Cryptology, AFRICACRYPT 2014. Proc.: Lecture Notes in Computer Science (LNCS 8469)*, Marrakesh, Morocco, pp. 336–355, 2014.
[23] Y. Zhang, Y. Hu and M. Jiang, "Lattice- based sequential aggregate signatures with lazy verification," *the Journal of China Universities of Posts and Telecommunications*, vol. 22, no. 6, pp. 36–44, 2015.
[24] G. Xu, Y. Cao, S. Xu, K. Xiao, X. Liu *et al.,* "A novel post-quantum blind signature for log system in blockchain," *Computer Systems Science and Engineering*, vol. 41, no. 3, pp. 945–958, 2022.
[25] A. Alghafis, H. M. Waseem, M. Khan and S. S. Jamal, "A hybrid cryptosystem for digital contents confidentiality based on rotation of quantum spin states," *Physica A: Statistical Mechanics and its Applications*, vol. 554(C), pp. 123908, 2020.
[26] H. M. Waseem, A. Alghafis and M. Khan, "An efficient public Key cryptosystem based on dihedral group and quantum spin states," *IEEE Access*, vol. 8, pp. 71821–71832, 2020.
[27] A. Alghafis, H. M. Waseem, M. Khan, S. S. Jamal, M. Amin *et al.,* "A novel digital contents privacy scheme based on quantum harmonic oscillator and schrodinger paradox," *Wireless Networks*, 2020. https://doi.org/10.1007/s11276-020-02363-7.

[28] M. Finiasz, "Nouvelles constructions utilisant des codes correcteurs derreurs en cryptographie 'a clef publique," Ph.D dissertation, INRIA C Ecole Polytechnique, France, 2004.
[29] N. Sendrier, "Code-based one way functions," 2007. [Online]. Available: http://ecrypt-ss07.rhul.ac.uk/Slides/Thursday/sendrier-samos07.pdf.
[30] R. Overbeck and N. Sendrier, "Code based cryptography," in D. J. Bernstein, J. Buchmann, E. Dahmen, (Eds.), *Post-Quantum Cryptography*, Berlin, Heidelberg, Germany: Springer, pp. 95–145, 2009.
[31] M. Finiasz and N. Sendrier, "Security bounds for the design of code-based cryptosystems," in *Advances in Cryptology, ASIACRYPT 2009. Proc.: Lecture Notes in Computer Science (LNCS 5912)*, Tokyo, Japan, pp. 88–105, 2009.