Research Article

A Fully Adaptively Secure Threshold Signature Scheme Based on Dual-Form Signatures Technology

Yuanju Gan

Faculty of Mathematics and Computer Science, Guangdong Ocean University, Zhanjiang 524088, China

Correspondence should be addressed to Yuanju Gan; ganyj66@163.com

Received 31 December 2020; Revised 6 February 2021; Accepted 19 February 2021; Published 9 March 2021

Academic Editor: Guodong Ye

Copyright © 2021 Yuanju Gan. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In (t, n) threshold signature schemes, any subset of t participants out of n can produce a valid signature, but any fewer than t participants cannot. Meanwhile, a threshold signature scheme should remain robust and unforgeable against up to $t - 1$ corrupted participants. This nonforgeability property is that even an adversary breaking into up to $t - 1$ participants should be unable to generate signatures on its own. Existential unforgeability against adaptive chosen message attacks is widely considered as a standard security notion for digital signature, and threshold signature should also follow this accordingly. However, there are two special attack models in a threshold signature scheme: one is the static corruption attack and the other is the adaptive corruption attack. Since the adaptive corruption model appears to better capture real threats, designing and proving threshold signature schemes secure in the adaptive corruption model has been focused on in recent years. If a threshold signature is secure under adaptive chosen message attack and adaptive corruption attack, we say it is fully adaptively secure. In this paper, based on the dual pairing vector spaces technology, we construct a threshold signature scheme and use Gerbush et al.'s dual-form signatures technology to prove our scheme, which is fully adaptively secure in the standard model, and then compare it to other schemes in terms of the efficiency and computation.

1. Introduction

A simple communication model, in which there is a single pair, sender and receiver, has to be extended by allowing communication between groups. To meet the security requirements for such a model, group-oriented cryptography was invented by Desmedt [1]. Given a group of n participants. A (t, n) threshold signature is a cryptographic protocol that allows any subgroup of t participants ($t \leq n$) to collectively sign messages. On the other hand, any subgroup of $(t - 1)$ participants (or smaller) is unable to generate a valid signature for any message. A threshold signature is called *robust* if misbehaving participants (who do not follow the protocol) are unable to prevent the honest participants from a successful execution of the signature protocol.

Goldwasser et al. studied the security of signatures in [2]. They introduced a security notion called *existential unforgeability against chosen message attacks* (EUF-CMA) and analysed it using an appropriate security game. In the game, an adversary can get a public key (for signature verification)

and she can query a signature oracle a polynomial number of times for messages of her choice. She wins the game (and breaks the security) if she is able to come up with a valid signature σ^* for a message m^* that has not been queried before. There are two flavours of the game: nonadaptive and adaptive. In a nonadaptive EUF-CMA, the adversary queries the signature oracle before she gets the public key. In an adaptive EUF-CMA, she gets the public key first and then she uses the signature oracle.

In group-oriented cryptography, there are two basic adversarial models: static and adaptive. In the static model, the adversary fixes a subset of participants that she wants to corrupt before the protocol starts [3, 4]. In the adaptive model, the adversary waits for an appropriate time during the protocol execution. She collects information from a protocol run and corrupts a specific subset of participants to maximise her chances of success [5–7]. Clearly, the adaptive adversary is strictly stronger than the static one. Besides, the adaptive model reflects better real-life adversaries and therefore it has attracted more attention from researchers.

Needless to say, it is a more challenging task to design threshold signatures with EUF-aCMA security under the adaptive adversarial model (than under the static model).

In 2004, Abe and Fehr in their work [8] proposed threshold signatures secure against adaptive corruptions using the universal composability framework. Although reduced, there is still a need for interactions. Wang et al. [7] apply the Waters signatures [9] to construct threshold signatures secure against adaptive adversaries in the standard model. Their signatures still require interactions via secure point-to-point channels.

Libert and Yung [10] used the Lewko-Waters identity-based encryption [11] and bilinear mapping over groups of composite order to design threshold signatures secure against adaptive adversaries, where there is no need for interactions. However, Freeman [12] argued that, for the same security level, the Libert-Yung signatures are roughly 50 times slower than signatures in prime-order groups. Hwang et al. in their work [13] constructed (t, n) threshold directed signatures that are existentially unforgeable under the chosen message attack under the assumption that the computational Diffie-Hellman problem is intractable. However, the security claims hold only when the adversary is static. Raman et al. in their work [14] proposed threshold proxy signatures based on the RSA assumptions. Both constructions [13, 14] are not secure against adaptive corruptions.

Libert et al. [15] employ the Pedersen distributed key generation protocol [16] to design two variants of threshold signatures that are adaptively secure against a static adversary. The efficiency of the first scheme is high, but the security proof is done in the random-oracle model. The second scheme is proven secure in the standard model using the Groth-Sahai arguments [17]. Unfortunately, the time price is low efficiency; for example, the time required for a partial signature is proportional to the length of the block message. Harn and Wang [18] have studied threshold signatures using the Chinese remainder theorem. Their designs are secure against adaptively chosen message attacks and static corruptions in the random-oracle model. Recently, Assidi et al. [19] have proposed an efficient code-based threshold ring signature. However, their scheme cannot withstand the adaptive corruptions attacks.

It turns out that the dual pairing vector space (DPVS) technique offers a powerful tool for constructing cryptographic algorithms such as identity-based encryption (IBE) and attribute-based encryption (ABE). We show that the technique can also be employed to design threshold signatures. The main contribution of our paper is the construction of (t, n) threshold signatures that are EUF-CMA secure under adaptive corruption attacks in the standard model. Our construction is based on the Lewko identity-based encryption [20] and the Okamoto-Takashima DPVS technique [21, 22]. The security proof, however, applies dual-form signature arguments developed by Gerbush et al. in [23].

2. Preliminaries

Given a set S . We denote by $x \xleftarrow{R} S$ the fact that x is drawn randomly and uniformly from S . $|S|$ denotes the set

cardinality and $|x|$ is the length of x (or the number of bits in its binary form). If $\mathbb{N}$ is the set of natural numbers, then 1^k denotes the string of k ones; $k \in \mathbb{N}$.

2.1. Bilinear Maps. Given two cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p , a bilinear map $e(\cdot, \cdot)$ is a function $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ such that, for any generator $g \in \mathbb{G}$ and any integers $\alpha, \beta \in \mathbb{Z}_p$, the map satisfies the following properties [24]:

- (i) Bilinearity: $e(g^\alpha, g^\beta) = e(g, g)^{\alpha\beta}$
- (ii) Nondegeneracy: $e(g, g) \neq 1_{\mathbb{G}_T}$ unless $g = 1_{\mathbb{G}}$
- (iii) Computability: there exists an efficient algorithm that computes $e(g_1, g_2)$ for $\forall g_1, g_2 \in \mathbb{G}$

Note that $\mathbb{Z}_p$ is the set of integers modulo p . Let us introduce the notation that will be used in the paper. Given a vector $\mathbf{w} = (w_1, \dots, w_\ell) \in \mathbb{Z}_p^\ell$ and an integer $a \in \mathbb{Z}_p$, we define the following shorthand notations:

- (i) $g^{\mathbf{w}} = (g^{w_1}, \dots, g^{w_\ell})$, where $g \in \mathbb{G}$
- (ii) $\mathbf{w}^a = (w_1^a, \dots, w_\ell^a)$
- (iii) $g^{w_1} g^{w_2} \dots g^{w_n} = g^{w_1 + w_2 + \dots + w_n}$, where $\mathbf{w}_i \in \mathbb{Z}_p^\ell$ for $i = 1, 2, \dots, n$ and $g \in \mathbb{G}$
- (iv) $\prod_{i=1}^n \mathbf{w}_i = (\prod_{i=1}^n w_{i,1}, \prod_{i=1}^n w_{i,2}, \dots, \prod_{i=1}^n w_{i,\ell})$, where $\mathbf{w}_i = (w_{i,1}, w_{i,2}, \dots, w_{i,\ell}) \in \mathbb{Z}_p^\ell$
- (v) $e_\ell(g^{\mathbf{v}}, g^{\mathbf{w}}) = \prod_{i=1}^\ell e(g^{v_i}, g^{w_i}) = e(g, g)^{\langle \mathbf{v}, \mathbf{w} \rangle}$, where $\mathbf{v}, \mathbf{w} \in \mathbb{Z}_p^\ell$, e is a bilinear map and $g \in \mathbb{G}$

A group generation algorithm $\mathcal{GG}(\Lambda)$ takes a requested security level parameter Λ and returns $(\mathbb{G}, \mathbb{G}_T, p, e(\cdot, \cdot), g)$, where p is a large prime and g is a generator of $\mathbb{G}$.

2.2. Dual Pairing Vector Spaces. Given two random bases $\mathbb{B} = (\mathbf{b}_1, \dots, \mathbf{b}_\ell)$ and $\mathbb{B}^* = (\mathbf{b}_1^*, \dots, \mathbf{b}_\ell^*)$ over $\mathbb{Z}_p^\ell$, the bases $\mathbb{B}$ and $\mathbb{B}^*$ are called *dual orthonormal* if $\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle = 0$ when $i \neq j$ and $\langle \mathbf{b}_i, \mathbf{b}_i^* \rangle = \psi$ when $i = 1, 2, \dots, \ell$, where $\psi \in \mathbb{Z}_p$. Note that, for a generator g of $\mathbb{G}$, the following relation holds:

$$e_\ell(g^{b_i}, g^{b_j^*}) = 1_{\mathbb{G}_T}, \quad \text{as long as } i \neq j. \quad (1)$$

$\text{Dual}(\mathbb{Z}_p^\ell)$ denotes the set of dual orthonormal bases [20, 21, 25].

2.3. Subspace Intractability Assumption. Given a group generator $\mathcal{GG}(\Lambda)$ and a positive integer $k \leq (\ell/3)$, the subspace assumption is defined as follows [20]. For a given security level Λ , generate the parameters of an instance at random:

$$(\mathbb{G}, \mathbb{G}_T, p, e(\cdot, \cdot), g) \xleftarrow{R} \mathcal{GG}(\Lambda),$$

$$(\mathbb{B}, \mathbb{B}^*) \xleftarrow{R} \text{Dual}(\mathbb{Z}_p^\ell), \quad (2)$$

$$\eta, \beta, \tau_1, \tau_2, \tau_3, \mu_1, \mu_2, \mu_3 \xleftarrow{R} \mathbb{Z}_p.$$

Calculate the following collection of vectors:

$$\begin{aligned}
U_i &= g^{\mu_i \mathbf{b}_i + \mu_2 \mathbf{b}_{k+i} + \mu_3 \mathbf{b}_{2k+i}}, \\
V_i &= g^{\tau_1 \eta \mathbf{b}_i^* + \tau_2 \beta \mathbf{b}_{k+i}^*}, \\
W_i &= g^{\tau_1 \eta \mathbf{b}_i^* + \tau_2 \beta \mathbf{b}_{k+i}^* + \tau_3 \mathbf{b}_{2k+i}^*},
\end{aligned} \tag{3}$$

where $i = 1, 2, \dots, k$. Then calculate a vector D such that

$$D = (\mathbb{G}, \mathbb{G}_T, p, e(\cdot, \cdot), g, g^{\mathbf{b}_1}, g^{\mathbf{b}_2}, \dots, g^{\mathbf{b}_{2k}}, g^{\mathbf{b}_{3k+1}}, \dots, g^{\mathbf{b}_\ell}, g^{\eta \mathbf{b}_1^*}, \dots, g^{\eta \mathbf{b}_k^*}, g^{\beta \mathbf{b}_{k+1}^*}, \dots, g^{\beta \mathbf{b}_{2k}^*}, g^{\mathbf{b}_{2k+1}^*}, \dots, g^{\mathbf{b}_\ell^*}, U_1, U_2, \dots, U_k, \mu_3). \tag{4}$$

We assume that, for any probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, the advantage in distinguishing vectors

V_i from vectors W_i , where $i = 1, 2, \dots, k$, is negligible in the security parameter Λ or

$$\text{ADV}_{\mathcal{G}, \mathcal{A}}(\Lambda) = |P[\mathcal{A}(D, V_1, \dots, V_k) = 1] - P[\mathcal{A}(D, W_1, \dots, W_k) = 1]| < \frac{1}{\text{poly}(\Lambda)}. \tag{5}$$

3. Building Blocks

3.1. Target Collision Resistant Hash Functions. Given a security parameter Λ , a $\ell(\Lambda)$ -bit key k , a group $\mathbb{G}$ of prime order $p(\Lambda)$, and a family of keyed hash functions $T = \{T_k: \{0, 1\}^* \rightarrow \mathbb{Z}_{p(\Lambda)}\}$ that is indexed by both the key k

and the security parameter, the family T is target collision resistant (TCR) if, for any PPT adversary $\mathcal{H}$, her advantage in finding a collision is negligible in the security parameter Λ or in other words (see also [26]):

$$\text{Adv}_{\mathcal{H}}^{\text{TCR}}(\Lambda) = \Pr \left[k \xleftarrow{R} \{0, 1\}^{\ell(\Lambda)}; c_1 \xleftarrow{R} \{0, 1\}^*; c_2 \xleftarrow{R} \mathcal{H}(k, c_1): T_k(c_1) = T_k(c_2) \wedge c_1 \neq c_2 \right] < \frac{1}{\text{poly}(\Lambda)}. \tag{6}$$

TCR is a weaker requirement of collision resistance than the standard collision resistance of hash functions without keys. Therefore, it is a better match for the hashing used in practice.

3.2. Signatures. We recall the definition of digital signatures and their security models.

Definition 1. A signature scheme (SS) is a triplet $\langle \text{KeyGen}, \text{Sign}, \text{Verify} \rangle$ of PPT algorithms described as follows:

- (i) $\text{KeyGen}(\Lambda)$: for a required security level Λ , the algorithm returns a pair of keys (pk, sk) of appropriate length. The key pk is public (for verification), while the key sk is secret (for signing)
- (ii) $\text{Sign}(\text{sk}, m)$: for an input, the secret key sk , and a message m , the algorithm returns a signature σ of the message m
- (iii) $\text{Verify}(\text{pk}, m, \sigma)$: for an input, the public key pk , the message m , and a string σ , the algorithm outputs **1** if σ is a valid signature of m . Otherwise, it returns **0**

The adaptive chosen message attack (aCMA) model is stronger than the nonadaptive chosen message attack

(naCMA) model. So, we only think about the EUF aCMA game.

3.3. Dual-Form Signatures. In 2012, Gerbush et al. [23] proposed the concept of dual-form signatures (DFSs), which is a useful framework for proving the security (existential unforgeability) of signatures from static assumptions. A dual-form signature is defined by the following algorithms [23]:

- (i) $\text{KeyGen}(\Lambda)$: generates a public key pk and a private key sk for a security parameter Λ
- (ii) $\text{Sign}_A(\text{sk}, m)$: returns a signature σ for a message m and a secret key sk
- (iii) $\text{Sign}_B(\text{sk}, m)$: outputs a signature σ for a message m and a secret key sk
- (iv) $\text{Verify}(\text{pk}, m, \sigma)$: outputs TRUE if the signature σ is valid for the message m and the public key pk . Otherwise, it returns FALSE

As one can see, dual-form signatures apply two variants of the signature algorithm, Sign_A and Sign_B . Both algorithms generate valid signatures. This means that they pass the verification algorithm Verify no matter which signature algorithm has been used. However, only one signing

algorithm will be used in the real scheme. The other signature algorithm will be used in the security proof/game. To prove the security of dual-form signatures, Gerbush et al. [23] gave a general proof method that works as follows.

In a security game, forgeries can be categorised into two (disjoint) types: I and II. They correspond to the signatures obtained by Sign_A and Sign_B , respectively. The security proof involves a sequence of transformations of queries obtained from both signature variants Sign_A and Sign_B . In the proof, an adversary is asked to produce a valid signature that can be of a specific form. The following list details the required properties:

A-I matching: an adversary is only given access to the oracle Sign_A . It is expected that it is harder for her to produce Type II forgeries

B-II matching: an adversary is only given access to the oracle Sign_B . It is expected that it is harder for her to produce Type I forgeries

Dual-oracle invariance: an adversary is given access to both oracles Sign_A and Sign_B . It is expected that the adversary has the same probability of success to generate Type I and Type II forgeries

A dual-form signature is secure if it satisfies the three properties listed above.

In [23], the author shows that, by the A-I matching property, it might have a noticeable probability ϵ of producing a Type I forgery but has only a negligible probability of producing any other kind of forgery, and show that ϵ must also be negligible. By the dual-oracle invariance property, the probability of producing a Type I forgery will be close to ϵ if gradually replacing the signing algorithm with Sign_B , one signature at a time. Once all of the signatures the attacker receives are from Sign_B , the B-II matching property implies that the probability of producing a Type I forgery must be negligible in the security parameter.

If $\langle \text{Setup}, \text{Sign}_A, \text{Sign}_B, \text{Verify} \rangle$ is a secure dual-form signature, then $\langle \text{Setup}, \text{Sign}_A, \text{Verify} \rangle$ is existentially unforgeable under an adaptive chosen message attack [23]. This proven technique has a great potential to be a useful tool in the analysis of other cryptographic primitives that use digital signatures.

3.4. Threshold Signatures. A threshold signature Φ consists of the following five algorithms $\langle \text{Setup}, \text{Share-sign}, \text{Share-verify}, \text{Combine}, \text{Verify} \rangle$.

(i) $\text{Setup}(\Lambda, t, n)$ takes as input a security parameter $\Lambda \in \mathbb{Z}$, a requested threshold t , and the total number of participants n , where $1 \leq t \leq n$. It outputs a set of parameters (PP, PK, SK, VK) , where PP is the system public parameter, PK is the public key, $SK = (SK_1, \dots, SK_n)$ is the vector of private keys, and $VK = (VK_1, \dots, VK_n)$ is the corresponding verification key vector. The private key SK_i is given to the i^{th} participant via a secure channel ($i = 1, 2, \dots, n$).

(ii) $\text{Share-sign}(P_i, SK_i, \Gamma, M)$ is a randomised algorithm executed by P_i . It takes the private key SK_i , a

message M , and a subset $\Gamma \subset \{1, \dots, n\}$ of size $|\Gamma| = t$ participants who want to sign a message M . It outputs a signature share u_i .

- (iii) $\text{Share-verify}(P_i, \Gamma, VK_i, u_i, M)$ is a deterministic algorithm. It takes as input an identity of $P_i \in \Gamma$, its verification key VK_i , its signature share u_i , and a message M . It outputs valid or invalid depending on whether u_i is considered to be a valid share or not.
- (iv) $\text{Combine}(PK, M, \Gamma, \{(P_i, u_i)\}_{P_i \in \Gamma})$ takes as input the public key PK , a message M , and a subset $\Gamma \subset \{1, \dots, n\}$ of size $|\Gamma| = t$ with pairs $\{(P_i, u_i)\}$ such that $P_i \in \Gamma$ and u_i is a signature share. This algorithm outputs either a full signature S or $\perp$ if the set contains invalid signature shares.
- (v) $\text{Verify}(PK, M, S)$ is a deterministic algorithm that takes as input a message M , the public key PK , and a signature S . This algorithm outputs valid or invalid depending on whether S is a valid signature or not.

We assume that an adversary $\mathcal{A}$ can corrupt up to $t - 1$ participants (from all n participants). $\mathcal{A}$ can learn all the information stored by the corrupted participants and can listen to all their communication. We consider two types of adversaries: static and adaptive. A static adversary corrupts participants at the beginning of a protocol. An adaptive adversary corrupts participants during protocol execution and chooses the best time so it maximises her chances of compromising the security of noncorrupted participants. As the adaptive adversary captures real-life threats better, we are going to consider threshold signatures that guarantee EUF-aCMA security under adaptive corruption attacks. Consequently, the other case of static corruption attacks is going to be ignored in this work.

Given an adaptive adversary and a threshold signature Φ , to evaluate the security of the signature in this setting, we need to develop an appropriate security notion. Fortunately, we can use the notion formulated by Libert and Yung in their work [LY13].

Definition 2. A threshold signature scheme Φ is existentially unforgeable under adaptively chosen message attacks and adaptive corruption attacks (EUF-aCMA-DCA) if no PPT adversary $\mathcal{A}$ has a nonnegligible advantage in the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

- (i) *Initialisation:* $\mathcal{C}$ runs $\text{Setup}(\Lambda, t, n)$.

$$PP, PK, SK, VK \leftarrow \text{Setup}(\Lambda, t, n). \quad (7)$$

The challenger sends public parameters to the adversary. In other words,

$$\mathcal{C} \longrightarrow \mathcal{A}: PP, PK, VK. \quad (8)$$

- (ii) *Queries:* $\mathcal{A}$ can adaptively invoke an arbitrary number of queries. Each query must be either a

corruption query or a signing query. The two types are defined below.

- (i) *Corruption query*: If $\mathcal{A}$ wants to corrupt the i th participant, then

$$\mathcal{C} \longrightarrow \mathcal{A}: \text{SK}_i. \quad (9)$$

No more than $t - 1$ private keys can be obtained by $\mathcal{A}$ in the whole game.

- (ii) *Partial signature query*: If the adversary $\mathcal{A}$ asks for partial signature on an arbitrary message M on behalf of a participant $P_i \in \Gamma$, then

$$\mathcal{C} \longrightarrow \mathcal{A}: u_i. \quad (10)$$

- (iii) *Signing query*: $\mathcal{A}$ asks for a signature on an arbitrary message M and then

$$\mathcal{C} \longrightarrow \mathcal{A}: S. \quad (11)$$

We have that S is a signature of M .

Forgery: $\mathcal{A}$ outputs a message M^* and a signature S^* . She wins if

- (i) M^* was never submitted to the partial signature query or signing query;
- (ii) $\mathcal{A}$ did not obtain more than $t - 1$ private key shares in the whole game;
- (iii) $\text{Verify}(\text{PK}, M^*, S^*) = \text{valid}$.

4. Construction

In this section, we present a threshold signature Φ based on dual pairing vector spaces. The security of the proposed scheme is proven under the subspace assumption. The algorithms of the signature are as follows.

Setup (Λ, t, n) : Given the parameters Λ , t , and n , the following steps are executed:

- (i) Compute group parameters by running the algorithm $\mathcal{GG}(\Lambda)$:

$$(\mathbb{G}, \mathbb{G}_T, p, e(\cdot, \cdot), g) \leftarrow \mathcal{GG}(\Lambda). \quad (12)$$

Select a hash function H , where $H: \{0, 1\}^* \longrightarrow \mathbb{Z}_p$ is a target collision-resistant hash function.

- (ii) Set $\ell = 6$ and sample a pair of random dual orthogonal bases.

$$(\mathbb{D}, \mathbb{D}^*) \xleftarrow{R} \text{Dual}(\mathbb{Z}_p^\ell). \quad (13)$$

Let $\mathbf{d}_1, \dots, \mathbf{d}_6$ denote the elements of $\mathbb{D}$ and let $\mathbf{d}_1^*, \dots, \mathbf{d}_6^*$ denote the elements of $\mathbb{D}^*$, where $\mathbf{d}_i, \mathbf{d}_i^* \in \mathbb{Z}_p^\ell$ and $\langle \mathbf{d}_i, \mathbf{d}_i^* \rangle = \psi$, for $i = 1, \dots, 6$.

- (iii) Choose random values $\alpha, \theta, \sigma \in \mathbb{Z}_p^*$ and a random polynomial $F(x) = \sum_{j=0}^{t-1} a_j x^j$ of degree $t - 1$ (where t is the threshold value and $a_j \in \mathbb{Z}_p$ for $j = 0, \dots, t - 1$, and $a_0 = \alpha$) and compute the partial decryption keys

$$\begin{aligned} \text{SK}_{i_1} &= g^{F(i)\theta \mathbf{d}_1^*}, \\ \text{SK}_{i_2} &= g^{r_{i_1} \theta \mathbf{d}_1^* - r_{i_1} \theta \mathbf{d}_2^* + r_{i_2} \sigma \mathbf{d}_3^* - r_{i_2} \sigma \mathbf{d}_4^*}, \end{aligned} \quad (14)$$

and verification key

$$\text{VK}_i = e_\ell(g^{\mathbf{d}_1}, \text{SK}_{i_1}) = e(g, g)^{F(i)\theta \psi}, \quad (15)$$

for $i = 1, \dots, n$, where $r_{i_1}, r_{i_2} \xleftarrow{R} \mathbb{Z}_p$.

- (iv) Publish the public parameters $\text{PP} = \{\mathbb{G}, \mathbb{G}_T, p, e(\cdot, \cdot), g, H(\cdot)\}$, public key $\text{PK} = \{Z = e(g, g)^{\alpha \theta \psi}, g^{\mathbf{d}_1}, g^{\mathbf{d}_2}, g^{\mathbf{d}_3}, g^{\mathbf{d}_4}\}$, and verification key $\text{VK} = (\text{VK}_1, \dots, \text{VK}_n)$ on the bulletin board. Partial private key $\text{SK}_i = (\text{SK}_{i_1}, \text{SK}_{i_2})$ is privately given to P_i , for $i = 1, \dots, n$.

Share-Sign $(P_i, \text{SK}_i, \Gamma, M)$: Denote the currently active set of t participants by Γ . P_i selects at random $k_i \in \mathbb{Z}_p$ and computes its partial signature (or signature share) $s_i = (u_i, \mathcal{K}_i)$ using SK_i according to

$$\begin{aligned} u_i &= \text{SK}_{i_1}^{\lambda_i + H(M)k_i} \text{SK}_{i_2}^{k_i}, \\ \mathcal{K}_i &= \text{VK}_i^{k_i}, \end{aligned} \quad (16)$$

where $\lambda_i = \prod_{j \in \Gamma, j \neq i} -j/(i - j)$, and then sends $s_i = (u_i, \mathcal{K}_i)$ to the combiner.

Share-Verify $(P_i, \Gamma, \text{VK}_i, u_i, M)$: Given P_i 's ($P_i \in \Gamma$) verification key VK_i and the partial signature $s_i = (u_i, \mathcal{K}_i)$ of the message M , the combiner verifies s_i by checking if

$$e_\ell(g^{\mathbf{d}_1} g^{\mathbf{d}_2} g^{\mathbf{d}_3} g^{\mathbf{d}_4}, u_i) \stackrel{?}{=} \text{VK}_i^{\lambda_i} \mathcal{K}_i^{H(M)}. \quad (17)$$

If the above equality holds, $s_i = (u_i, \mathcal{K}_i)$ is valid. Otherwise, the combiner declares that P_i is corrupted.

Combine $(\text{PK}, M, \Gamma, \{(P_i, s_i)\}_{P_i \in \Gamma})$:

Given the set Γ , the public key PK , the message M , and t valid shares $(\{(P_i, s_i)\}_{P_i \in \Gamma})$, the combiner computes the signature $\mathcal{S} = (S_1, S_2)$ as

$$\begin{aligned} S_1 &= \prod_{i \in \Gamma} u_i, \\ S_2 &= \prod_{i \in \Gamma} \mathcal{K}_i. \end{aligned} \quad (18)$$

Verify $(\text{PK}, M, \mathcal{S})$: For a signature $\mathcal{S} = (S_1, S_2)$ of a message M with respect to a public key PK , the

algorithm outputs valid (or 1) if the following equation holds:

$$e_\ell(g^{d_1} g^{d_2} g^{d_3} g^{d_4}, S_1) \stackrel{?}{=} ZS_2^{H(M)}. \quad (19)$$

Otherwise, it returns invalid (or 0).

4.1. Correctness. We are going to show that the two verification algorithms Share-Verify and Verify always accept the well-formed shares and the final signatures. The Share-Verify algorithm accepts the input if equation (17) holds. We start from the left-hand side of the equation and run through a sequence of transformations, so we arrive at the right-hand side of equation (17).

$$\begin{aligned} & e_\ell(g^{d_1} g^{d_2} g^{d_3} g^{d_4}, u_i) \\ &= e_\ell(g^{d_1} g^{d_2} g^{d_3} g^{d_4}, SK_{i_1}^{\lambda_i + H(M)k_i} SK_{i_2}^{k_i}) \\ &= e_\ell(g^{d_1} g^{d_2} g^{d_3} g^{d_4}, SK_{i_1}^{\lambda_i + H(M)k_i}) e_\ell(g^{d_1} g^{d_2} g^{d_3} g^{d_4}, SK_{i_2}^{k_i}) \\ &= e_\ell(g^{d_1} g^{d_2} g^{d_3} g^{d_4}, g^{F(i)\theta d_1^* \lambda_i + H(M)k_i}) e_\ell(g^{d_1} g^{d_2} g^{d_3} g^{d_4}, SK_{i_2}^{k_i}) \\ &= e(g, g)^{F(i)\theta \psi \lambda_i + H(M)k_i} e_\ell(g^{d_1} g^{d_2} g^{d_3} g^{d_4}, SK_{i_2}^{k_i}) \\ &= VK_{i_1}^{\lambda_i} \mathcal{K}_i^{H(M)} e_\ell(g^{d_1} g^{d_2} g^{d_3} g^{d_4}, SK_{i_2}^{k_i}) \\ &= VK_{i_1}^{\lambda_i} \mathcal{K}_i^{H(M)} e_\ell\left(g^{d_1} g^{d_2} g^{d_3} g^{d_4}, g^{(r_{i_1} \theta d_1^* - r_{i_1} \theta d_2^* + r_{i_2} \sigma d_3^* - r_{i_2} \sigma d_4^*) k_i}\right) \\ &= VK_{i_1}^{\lambda_i} \mathcal{K}_i^{H(M)} e(g, g)^{(r_{i_1} \theta d_1^* - r_{i_1} \theta d_2^* + r_{i_2} \sigma d_3^* - r_{i_2} \sigma d_4^*) k_i} \\ &= VK_{i_1}^{\lambda_i} \mathcal{K}_i^{H(M)} e(g, g)^{(r_{i_1} \theta \psi - r_{i_1} \theta \psi + r_{i_2} \sigma \psi - r_{i_2} \sigma \psi) k_i} \\ &= VK_{i_1}^{\lambda_i} \mathcal{K}_i^{H(M)}. \end{aligned} \quad (20)$$

The Verify algorithm uses equation (19) to check if the signatures are valid. In a similar vein, we start from the left-hand side of equation (19) and show how to arrive at the right-hand side of the equation.

$$\begin{aligned} S_1 &= \prod_{i \in \Gamma} u_i \\ &= \prod_{i \in \Gamma} SK_{i_1}^{\lambda_i + H(M)k_i} SK_{i_2}^{k_i} \\ &= \prod_{i \in \Gamma} SK_{i_1}^{\lambda_i + H(M)k_i} \cdot \prod_{i \in \Gamma} SK_{i_2}^{k_i} \\ &= \prod_{i \in \Gamma} SK_{i_1}^{\lambda_i} SK_{i_1}^{H(M)k_i} \cdot \prod_{i \in \Gamma} SK_{i_2}^{k_i} \\ &= \prod_{i \in \Gamma} (g^{F(i)\theta d_1^*})^{\lambda_i} \prod_{i \in \Gamma} SK_{i_1}^{H(M)k_i} \prod_{i \in \Gamma} SK_{i_2}^{k_i} \\ &= g^{\left(\sum_{i \in \Gamma} F(i)\lambda_i\right) \theta d_1^*} \prod_{i \in \Gamma} SK_{i_1}^{H(M)k_i} \prod_{i \in \Gamma} SK_{i_2}^{k_i} \\ &= g^{\alpha \theta d_1^*} \prod_{i \in \Gamma} SK_{i_1}^{H(M)k_i} \prod_{i \in \Gamma} (g^{r_{i_1} k_i \cdot \theta d_1^* - r_{i_1} k_i \cdot \theta d_2^* + r_{i_2} k_i \cdot \sigma d_3^* - r_{i_2} k_i \cdot \sigma d_4^*}) \\ &= g^{\alpha \theta d_1^*} g^{R_1 \theta d_1^* - R_1 \theta d_2^* + R_2 \sigma d_3^* - R_2 \sigma d_4^*} \prod_{i \in \Gamma} SK_{i_1}^{H(M)k_i}, \end{aligned} \quad (21)$$

where $R_1 = \sum_{i \in \Gamma} r_{i_1} k_i$, $R_2 = \sum_{i \in \Gamma} r_{i_2} k_i$, and let $\mathcal{W} = e_\ell(g^{d_1} g^{d_2} g^{d_3} g^{d_4}, S_1)$, so we have that

$$\begin{aligned} \mathcal{W} &= e_\ell\left(g^{d_1} g^{d_2} g^{d_3} g^{d_4}, g^{\alpha \theta d_1^*} g^{R_1 \theta d_1^* - R_1 \theta d_2^* + R_2 \sigma d_3^* - R_2 \sigma d_4^*} \prod_{i \in \Gamma} SK_{i_1}^{H(M)k_i}\right) \\ &= e(g, g)^{\alpha \theta \psi} e_\ell(g^{d_1} g^{d_2} g^{d_3} g^{d_4}, g^{R_1 \theta d_1^* - R_1 \theta d_2^* + R_2 \sigma d_3^* - R_2 \sigma d_4^*}) \prod_{i \in \Gamma} e(g, g)^{\theta \psi H(M)F(i)k_i} \\ &\quad \cdot Z \cdot e_\ell(g, g)^{R_1 \theta d_1^* d_1 - R_1 \theta d_2^* d_2 + R_2 \sigma d_3^* d_3 - R_2 \sigma d_4^* d_4} \prod_{i \in \Gamma} \mathcal{K}_i^{H(M)} \\ &\quad \cdot Z \cdot e(g, g)^{R_1 \theta \psi - R_1 \theta \psi + R_2 \sigma \psi - R_2 \sigma \psi} S_2^{H(M)} \\ &\quad \cdot Z \cdot S_2^{H(M)}. \end{aligned} \quad (22)$$

5. Security

Theorem 1. *Given our threshold signature $\Phi = \langle \text{Setup}, \text{Share-sign}, \text{Share-Verify}, \text{Combine}, \text{and Verify} \rangle$, assume that the subspace intractability assumption holds; then the signature scheme Φ is EUF-aCMA-DCA secure (or can withstand existential forgeries under adaptive chosen message and adaptive corruption attacks).*

We prove the theorem using the hybrid argument for a sequence of games similar to [23]. Like [23], our proof makes use of normal signatures and semifunctional semisignatures.

Type I signatures are normal signatures of S_1 :

$$S_1 = g^{(\alpha + R_3 H(M)) \theta d_1^*} g^{R_1 \theta d_1^* - R_1 \theta d_2^* + R_2 \sigma d_3^* - R_2 \sigma d_4^*}, \quad (23)$$

where $R_1, R_2 \in \mathbb{Z}_p$, $R_3 = \sum_{i \in \Gamma} F(i)k_i$.

Type II signatures are semisignatures of S_1 :

$$S_1 = g^{(\alpha + R_3 H(M))\theta \mathbf{d}_1^*} g^{R_1 \theta \mathbf{d}_2^* - R_1 \theta \mathbf{d}_2^* + R_2 \theta \mathbf{d}_3^* - R_2 \theta \mathbf{d}_4^* + z_5 \mathbf{d}_5^* + z_6 \mathbf{d}_6^*}, \quad (24)$$

where $R_1, R_2, z_5, z_6 \in \mathbb{Z}_p$, $R_3 = \prod_{i \in \Gamma} F(i)k_i \theta$.

We apply the dual-form signature technique to prove the security of our threshold signatures. We start from a real security game, denoted by Game_0 followed by the sequence $\text{Game}_1, \dots, \text{Game}_q$ and $\text{Game}_{\text{final}}$.

- (i) Game_0 is identical to the EUF-aCMA-DCA game, an adversary produces a Type I signature as a forgery, and the advantage of adversary $\mathcal{A}$ is defined accordingly. Game_0 is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ as follows.

Initialisation: $\mathcal{C}$ runs $\text{Setup}(\Lambda, t, n)$ to get $(\text{PP}, \text{PK}, \text{SK}, \text{VK})$, where

$$\begin{aligned} \text{PP} &= (G, G_T, p, e(\cdot, \cdot), g), \\ \text{PK} &= (Z = e(g, g)^{\alpha \theta \psi}, g^{\mathbf{d}_1}, g^{\mathbf{d}_2}, g^{\mathbf{d}_3}, g^{\mathbf{d}_4}, H(\cdot)), \\ \text{SK} &= (\text{SK}_1, \dots, \text{SK}_n) \text{ with } \text{SK}_i = (\text{SK}_{i1}, \text{SK}_{i2}), \\ \text{VK} &= (\text{VK}_1, \dots, \text{VK}_n). \end{aligned} \quad (25)$$

Queries: $\mathcal{A}$ adaptively makes a number q of queries.

Each query can be either a corruption one or a signing one. $\mathcal{A}$ cannot make more than $t-1$ corruption queries.

$\mathcal{A}$ gets also valid signatures for messages of her choice.

Forgery: $\mathcal{A}$ outputs $S^* = (S_1^*, S_2)$ for a message M^* that has not been queried.

$\mathcal{C}$ wins the game if $\text{Verify}(\text{PK}, M^*, S^*) = 1$.

- (ii) Game_j ($1 \leq j \leq q$) is like Game_{j-1} , except that the challenger $\mathcal{C}$ answers the first J signing query by returning Type II signatures, and the last $q-J$ signing queries are answered using Type I signatures. The process is as follows.

Initialisation: $\mathcal{C}$ runs $\text{Setup}(\Lambda, t, n)$ to get $(\text{PP}, \text{PK}, \text{SK}, \text{VK})$, where

$$\begin{aligned} \text{PP} &= (G, G_T, p, e(\cdot, \cdot), g), \\ \text{PK} &= (Z = e(g, g)^{\alpha \theta \psi}, g^{\mathbf{d}_1}, g^{\mathbf{d}_2}, g^{\mathbf{d}_3}, g^{\mathbf{d}_4}, H(\cdot)), \\ \text{SK} &= (\text{SK}_1, \dots, \text{SK}_n) \text{ with } \text{SK}_i = (\text{SK}_{i1}, \text{SK}_{i2}), \\ \text{VK} &= (\text{VK}_1, \dots, \text{VK}_n). \end{aligned} \quad (26)$$

Queries: $\mathcal{A}$ adaptively makes a number q of queries.

Each query can be either a corruption one or a signing one. $\mathcal{A}$ cannot make more than $t-1$ corruption queries.

$\mathcal{A}$ gets Type II signatures of the first J signing queries and gets Type I signatures of the last $q-J$ signing queries for messages of her choice.

Forgery: $\mathcal{A}$ outputs $S^* = (S_1^*, S_2)$ for a message M^* that has not been queried.

$\mathcal{A}$ wins the game if $\text{Verify}(\text{PK}, M^*, S^*) = 1$ and the probability is the same for the Type I and Type II forgeries.

- (iii) $\text{Game}_{\text{final}}$ is identical to Game_q , except that the challenger $\mathcal{C}$ answers the corruption query by returning the private key shares which contain random $\mathbf{d}_5^*, \mathbf{d}_6^*$, signing the query by returning Type II signatures, and needs to output a forgery of the same type. The process is as follows.

Initialisation: $\mathcal{C}$ runs $\text{Setup}(\Lambda, t, n)$ to get $(\text{PP}, \text{PK}, \text{SK}, \text{VK})$, where

$$\begin{aligned} \text{PP} &= (G, G_T, p, e(\cdot, \cdot), g), \\ \text{PK} &= (Z = e(g, g)^{\alpha \theta \psi}, g^{\mathbf{d}_1}, g^{\mathbf{d}_2}, g^{\mathbf{d}_3}, g^{\mathbf{d}_4}, H(\cdot)), \\ \text{SK} &= (\text{SK}_1, \dots, \text{SK}_n) \text{ with } \text{SK}_i = (\text{SK}_{i1}, \text{SK}_{i2}), \\ \text{VK} &= (\text{VK}_1, \dots, \text{VK}_n). \end{aligned} \quad (27)$$

Queries: $\mathcal{A}$ adaptively makes a number q of queries.

Each query can be either a corruption one or a signing one.

She cannot make more than $t-1$ corruption queries.

$\mathcal{A}$ gets the secret key $\text{SK}_i = (\text{SK}_{i1}, \text{SK}_{i2} g^{\tau_1 \mathbf{d}_5^* + \tau_2 \mathbf{d}_6^*})$ (where $\tau_1, \tau_2 \in \mathbb{Z}_p$) when she corrupts the participant i .

$\mathcal{A}$ gets Type II signatures of signing queries for messages of her choice.

Forgery: $\mathcal{A}$ outputs $S^* = (S_1^*, S_2)$ for a message M^* that has not been queried.

$\mathcal{A}$ wins the game if $\text{Verify}(\text{PK}, M^*, S^*) = 1$ and the forgery signature is Type I.

We will prove that Theorem 1 is true through a sequence of claims.

5.1. Claim 1. In Game_0 , adversary $\mathcal{A}$ has a negligible chance of outputting a Type II signature if the subspace assumption with $k = 2$ and $\ell = 6$ holds.

Proof. We construct an algorithm $\mathcal{D}$ that, on the input of an instance $(\mathbb{G}, \mathbb{G}_T, p, e(\cdot, \cdot), g, g^{\mathbf{b}_1}, \dots, g^{\mathbf{b}_4}, g^{\eta \mathbf{b}_1^*}, g^{\eta \mathbf{b}_2^*}, g^{\beta \mathbf{b}_3^*}, g^{\beta \mathbf{b}_4^*}, g^{\mathbf{b}_5^*}, g^{\mathbf{b}_6^*}, U_1 = g^{\mu_1 \mathbf{b}_1 + \mu_2 \mathbf{b}_3 + \mu_3 \mathbf{b}_5}, U_2 = g^{\mu_1 \mathbf{b}_2 + \mu_2 \mathbf{b}_4 + \mu_3 \mathbf{b}_6}, \mu_3, T_1, T_2)$, decides $T_1 = g^{\tau_1 \eta \mathbf{b}_1^* + \tau_2 \beta \mathbf{b}_3^*}, T_2 = g^{\tau_1 \eta \mathbf{b}_2^* + \tau_2 \beta \mathbf{b}_4^*}$ or $T_1 = g^{\tau_1 \eta \mathbf{b}_1^* + \tau_2 \beta \mathbf{b}_3^* + \tau_3 \mathbf{b}_5^*}, T_2 = g^{\tau_1 \eta \mathbf{b}_2^* + \tau_2 \beta \mathbf{b}_4^* + \tau_3 \mathbf{b}_6^*}$.

Initialisation: To compute the system public parameters and the participant's private key, $\mathcal{D}$ first selects a random invertible matrix $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \in \mathbb{Z}_p^{2 \times 2}$ and a TCR hash function $H: \{0, 1\}^* \rightarrow \mathbb{Z}_p$. The public parameter PP is $\{\mathbb{G}, \mathbb{G}_T, p, e(\cdot, \cdot), g, H\}$.

$\mathcal{D}$ defines the dual orthonormal bases $\mathbb{F} = (\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_4, \mathbf{f}_5, \mathbf{f}_6)$ and $\mathbb{F}^* = (\mathbf{f}_1^*, \mathbf{f}_2^*, \mathbf{f}_3^*, \mathbf{f}_4^*, \mathbf{f}_5^*, \mathbf{f}_6^*)$ by

$$\begin{aligned} \mathbf{f}_1 &= \eta \mathbf{b}_1^*, \\ \mathbf{f}_2 &= \eta \mathbf{b}_2^*, \\ \mathbf{f}_3 &= \beta \mathbf{b}_3^*, \\ \mathbf{f}_4 &= \beta \mathbf{b}_4^*, \\ \mathbf{f}_5 &= \mathbf{b}_5^*, \\ \mathbf{f}_6 &= \mathbf{b}_6^*, \\ \mathbf{f}_1^* &= \eta^{-1} \mathbf{b}_1, \\ \mathbf{f}_2^* &= \eta^{-1} \mathbf{b}_2, \\ \mathbf{f}_3^* &= \beta^{-1} \mathbf{b}_3, \\ \mathbf{f}_4^* &= \beta^{-1} \mathbf{b}_4, \\ \mathbf{f}_5^* &= \mathbf{b}_5, \\ \mathbf{f}_6^* &= \mathbf{b}_6. \end{aligned} \quad (28)$$

$\mathcal{D}$ sets $\mathbb{D} = \mathbb{F} \times \mathbb{A}$, $\mathbb{D}^* = \mathbb{F}^* \times \mathbb{A}^*$, where $\mathbb{A} = \begin{pmatrix} I_{4 \times 4} & 0 \\ 0 & A \end{pmatrix}$ and $\mathbb{A}^* = \begin{pmatrix} I_{4 \times 4} & 0 \\ 0 & (A')^{-1} \end{pmatrix}$ (A' denotes the transpose of A).

So, we have that $\mathbf{d}_i = \mathbf{f}_i, \mathbf{d}_i^* = \mathbf{f}_i^*$, for $i = 1, \dots, 4$, and $(\mathbf{d}_5, \mathbf{d}_6) = (\mathbf{f}_5, \mathbf{f}_6)A = (a_{11}\mathbf{f}_5 + a_{11}\mathbf{f}_6, a_{21}\mathbf{f}_5 + a_{22}\mathbf{f}_6)$, $(\mathbf{d}_5^*, \mathbf{d}_6^*) = (\mathbf{f}_5^*, \mathbf{f}_6^*)(A')^{-1}$. By Lemma 1 in [20], $\mathbb{D}$ and $\mathbb{D}^*$ are a pair of properly distributed dual orthonormal bases and reveal no information about A .

$\mathcal{D}$ picks random values $\alpha, \theta', \sigma' \in \mathbb{Z}_p$ and a random polynomial $F(x) \in \mathbb{Z}_p$ with degree $t-1$ and implicitly sets $\theta = \theta'\eta, \sigma = \sigma'\beta$, and $F(0) = \alpha$. $\mathcal{D}$ computes the public key as follows:

$$\begin{aligned} Z &= e_\ell(g^{\mathbf{b}_1}, g^{\eta \mathbf{b}_1^*})^{\alpha \theta'} \\ &= e(g, g)^{\alpha \theta' \mathbf{b}_1 \cdot \eta \mathbf{b}_1^*} \\ &= e(g, g)^{\alpha \theta' \eta (\eta^{-1} \mathbf{b}_1 \cdot \eta \mathbf{b}_1^*)} \\ &= e(g, g)^{\alpha \cdot \theta' \eta \mathbf{f}_1^* \cdot \mathbf{f}_1} \\ &= e(g, g)^{\alpha \theta \mathbf{d}_1^* \cdot \mathbf{d}_1}, \end{aligned} \quad (29)$$

$$\begin{aligned} g^{\mathbf{d}_1} &= g^{\eta \mathbf{b}_1^*}, \\ g^{\mathbf{d}_2} &= g^{\eta \mathbf{b}_2^*}, \\ g^{\mathbf{d}_3} &= g^{\beta \mathbf{b}_3^*}, \\ g^{\mathbf{d}_4} &= g^{\beta \mathbf{b}_4^*}. \end{aligned}$$

Since $(g^{\mathbf{b}_5^*})^{a_{11}} (g^{\mathbf{b}_6^*})^{a_{21}} = g^{a_{11}\mathbf{b}_5^* + a_{21}\mathbf{b}_6^*} = g^{a_{11}\mathbf{f}_5 + a_{21}\mathbf{f}_6} = g^{\mathbf{d}_5}$, we have that $g^{\mathbf{d}_5} = g^{a_{11}\mathbf{b}_5^* + a_{21}\mathbf{b}_6^*}$, and we also can get that $g^{\mathbf{d}_6} =$

$g^{a_{12}\mathbf{b}_5^* + a_{22}\mathbf{b}_6^*}$ by using the same method. $\mathcal{D}$ selects $r_{i_1}, r_{i_2} \xleftarrow{R} \mathbb{Z}_p$ (for $i = 1, \dots, n$) and computes P_i 's partial decryption key:

$$\begin{aligned} \text{SK}_{i_1} &= (g^{\mathbf{b}_1})^{F(i)\theta'} = g^{F(i)\eta\theta' \cdot \eta^{-1}\mathbf{b}_1} = g^{F(i)\theta \mathbf{d}_1^*}, \\ \text{SK}_{i_2} &= (g^{\mathbf{b}_1})^{r_{i_1}\theta'} (g^{\mathbf{b}_2})^{-r_{i_1}\theta'} (g^{\mathbf{b}_3})^{r_{i_2}\sigma'} (g^{\mathbf{b}_4})^{-r_{i_2}\sigma'} \\ &= g^{r_{i_1}\theta' \eta \mathbf{f}_1^* - r_{i_1}\theta' \eta \mathbf{f}_2^* + r_{i_2}\sigma' \beta \mathbf{f}_3^* - r_{i_2}\sigma' \beta \mathbf{f}_4^*} \\ &= g^{r_{i_1}\theta \mathbf{d}_1^* - r_{i_1}\theta \mathbf{d}_2^* + r_{i_2}\sigma \mathbf{d}_3^* - r_{i_2}\sigma \mathbf{d}_4^*}. \end{aligned} \quad (30)$$

Finally, $\mathcal{D}$ sends $\{\mathbb{G}, \mathbb{G}_T, p, e(\cdot, \cdot), g, H(\cdot)\}$, and $\text{PK} = \{Z, g^{\mathbf{d}_1}, g^{\mathbf{d}_2}, g^{\mathbf{d}_3}, g^{\mathbf{d}_4}\}$ to adversary $\mathcal{A}$.

Query. Since $\mathcal{D}$ knows all SK_i , it can perfectly answer corruption queries and signing queries.

$\mathcal{A}$ produces a forgery signature (M^*, S^*) , where $S^* = (S_1^*, S_2^*)$. Firstly $\mathcal{D}$ must check that $\text{Verify}(\text{PK}, M^*, S^*) = 1$; if not, then $\mathcal{D}$ will abort. In addition, $\mathcal{D}$ must verify that it has not received a Type I forgery signature. Notice that $\mathcal{D}$ needs the output of $\mathcal{A}$ to be a Type II forgery signature in order to defeat the subspace assumption with $k = 2$ and $l = 6$. To do this, $\mathcal{D}$ can check whether $e_\ell(g^{\mathbf{d}_5} g^{\mathbf{d}_6}, S_1^*) = 1_{\mathbb{G}_T}$ or not. If so, the forgery signature (M^*, S^*) is a Type I signature, and then $\mathcal{D}$ will abort. Otherwise, $\mathcal{D}$ uses the forgery signature to determine what subspace T_1, T_2 is in. Since $\mathcal{D}$ knows the value θ', α , it can simply check whether $e_\ell(T_1 T_2, S_1^*) / e_\ell(g^{(\alpha + R_3 H(M^*)) \mathbf{b}_1 \theta'}, T_1) = 1_{\mathbb{G}_T}$ or not. If so, $T_1 = g^{\tau_1 \eta \mathbf{b}_1^* + \tau_2 \beta \mathbf{b}_3^*}, T_2 = g^{\tau_1 \eta \mathbf{b}_2^* + \tau_2 \beta \mathbf{b}_4^*}$. Otherwise, $T_1 = g^{\tau_1 \eta \mathbf{b}_1^* + \tau_2 \beta \mathbf{b}_3^* + \tau_3 \mathbf{b}_5^*}, T_2 = g^{\tau_1 \eta \mathbf{b}_2^* + \tau_2 \beta \mathbf{b}_4^* + \tau_3 \mathbf{b}_6^*}$. $\square$

5.2. Claim 2. The adversary outputs a Type I forgery with negligibly different probabilities in Game_I and Game_{J+1} if the subspace assumption with $k = 2$ and $l = 6$ holds.

Proof. $\mathcal{D}$ begins by taking an instance $(\mathbb{G}, \mathbb{G}_T, p, e(\cdot, \cdot), g, g^{\mathbf{b}_1}, \dots, g^{\mathbf{b}_4}, g^{\eta \mathbf{b}_1^*}, g^{\eta \mathbf{b}_2^*}, g^{\beta \mathbf{b}_3^*}, g^{\beta \mathbf{b}_4^*}, g^{\mathbf{b}_5^*}, g^{\mathbf{b}_6^*}, U_1 = g^{\mu_1 \mathbf{b}_1 + \mu_2 \mathbf{b}_3 + \mu_3 \mathbf{b}_5}, U_2 = g^{\mu_1 \mathbf{b}_2 + \mu_2 \mathbf{b}_4 + \mu_3 \mathbf{b}_6}, \mu_3, T_1, T_2)$. $\mathcal{D}$'s task is to decide $T_1 = g^{\tau_1 \eta \mathbf{b}_1^* + \tau_2 \beta \mathbf{b}_3^*}, T_2 = g^{\tau_1 \eta \mathbf{b}_2^* + \tau_2 \beta \mathbf{b}_4^*}$ or $T_1 = g^{\tau_1 \eta \mathbf{b}_1^* + \tau_2 \beta \mathbf{b}_3^* + \tau_3 \mathbf{b}_5^*}, T_2 = g^{\tau_1 \eta \mathbf{b}_2^* + \tau_2 \beta \mathbf{b}_4^* + \tau_3 \mathbf{b}_6^*}$. Recall that $\mathcal{A}$ must obtain Type II signatures from the first J signing queries and Type I signatures at the last $q - J - 1$ queries. The J^{th} query will be a Type I signature or Type II signature depending on which subspace of T_1, T_2 is in.

Initialisation: To compute the public key, participants' partial decryption key, and the verification key, $\mathcal{D}$ first picks a random invertible matrix $A \in \mathbb{Z}_p^{2 \times 2}$ and a TCR hash function $H: \{0, 1\}^* \rightarrow \mathbb{Z}_p$. The public parameter PP is $\{\mathbb{G}, \mathbb{G}_T, p, e(\cdot, \cdot), g, H\}$; $\mathcal{D}$ defines the dual orthonormal bases $\mathbb{B} = \{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4, \mathbf{b}_5, \mathbf{b}_6\}$ and $\mathbb{B}^* = \{\mathbf{b}_1^*, \mathbf{b}_2^*, \mathbf{b}_3^*, \mathbf{b}_4^*, \mathbf{b}_5^*, \mathbf{b}_6^*\}$. $\mathcal{D}$ implicitly sets $\mathbb{D} = \mathbb{B} \times \begin{pmatrix} I_{4 \times 4} & 0 \\ 0 & A \end{pmatrix}$, $\mathbb{D}^* = \mathbb{B}^* \times \begin{pmatrix} I_{4 \times 4} & 0 \\ 0 & (A')^{-1} \end{pmatrix}$, $\theta = \eta$, and $\sigma = \beta$. So $\mathcal{D}$ can generate $g^{\mathbf{d}_1} = g^{\mathbf{b}_1}, g^{\mathbf{d}_2} = g^{\mathbf{b}_2}, g^{\mathbf{d}_3} = g^{\mathbf{b}_3}$, and $g^{\mathbf{d}_4} = g^{\mathbf{b}_4}$ and

also can generate $g^{\theta d_1^*} = g^{\eta b_1^*}$, $g^{\theta d_2^*} = g^{\eta b_2^*}$, $g^{\sigma d_3^*} = g^{\beta b_3^*}$, and $g^{\sigma d_4^*} = g^{\beta b_4^*}$. $\mathcal{D}$ picks a random value $\alpha, r \in \mathbb{Z}_p$ and a random polynomial $F(x) \in \mathbb{Z}_p$ with degree $t-1$ and sets $F(0) = \alpha$. $\mathcal{D}$ computes

$$Z = e_\ell(g^{b_1}, g^{\eta b_1^*})^\alpha = e(g, g)^{\alpha \theta d_1 \cdot d_1^*}, \quad (31)$$

as well as P_i 's partial decryption key

$$\begin{aligned} SK_{i_1} &= g^{\eta b_1^* F(i)} = g^{\theta d_1^* F(i)}, \\ SK_{i_2} &= g^{r \eta b_1^*} g^{-r \eta b_2^*} g^{r \beta b_3^*} g^{-r \beta b_4^*} = g^{r \theta d_1^* - r \theta d_2^* + r \sigma d_3^* - r \sigma d_4^*}, \end{aligned} \quad (32)$$

for $i = 1, \dots, n$. Finally, $\mathcal{D}$ sends PP, PK = $(Z, g^{d_1}, g^{d_2}, g^{d_3}, g^{d_4})$ to adversary $\mathcal{A}$.

Query: $\mathcal{D}$ can perfectly answer private key queries (since it knows SK_i) and answer $\mathcal{A}$'s signing queries depending on the value of $j \in \{1, \dots, q\}$.

Case $j < J$: to generate a signature share on a message M_j on behalf of $P_i, i \in \{1, \dots, n\}$, $\mathcal{D}$ chooses $k_i, \tau_5, \tau_6 \in_R \mathbb{Z}_p$ and computes a Type II signature share $u_i = SK_{i_1}^{\lambda_i + H(M)k_i} SK_{i_2}^{k_i} g^{\tau_5 b_5^*} g^{\tau_6 b_6^*}$.

Case $j > J$: in this case, $\mathcal{D}$ chooses $k_i \in_R \mathbb{Z}_p$ and computes a Type I signature share $u_i = SK_{i_1}^{\lambda_i + H(M)k_i} SK_{i_2}^{k_i}$.

Case $j = J$: in this case, $\mathcal{D}$ generates a signature share $u_i = SK_{i_1}^{\lambda_i + H(M)k_i} T_1^{k_i} T_2^{-k_i}$. It is easy to see that, in the situation $S_1 = \prod_{i \in \Gamma} u_i$ will be a Type II signature when $T_1 = g^{\tau_1 \eta b_1^* + \tau_2 \beta b_3^* + \tau_3 b_5^*}$, $T_2 = g^{\tau_1 \eta b_2^* + \tau_2 \beta b_4^* + \tau_3 b_6^*}$, or a Type I signature when $T_1 = g^{\tau_1 \eta b_1^* + \tau_2 \beta b_3^*}$, $T_2 = g^{\tau_1 \eta b_2^* + \tau_2 \beta b_4^*}$.

$\mathcal{A}$ sends a forgery signature (M^*, S^*) to $\mathcal{D}$. Firstly, $\mathcal{D}$ must check that $\text{Verify}(\text{PK}, M^*, S^*) = 1$; if not, then $\mathcal{D}$ will abort. In addition, $\mathcal{D}$ must identify which type of forgery signature it is. To do this, $\mathcal{D}$ will check whether $e_\ell(U_1 U_2, S_1^*) / e_\ell(g^{\alpha + R_3 H(M^*) \theta d_1^*}, U_1) = 1_{\mathbb{G}_T}$ or not. If so, the forgery signature (M^*, S^*) is a Type I signature. Otherwise, it is a Type II signature. At this stage, $\mathcal{D}$ returns 0 (meaning that it believes that $T_1 = g^{\tau_1 \eta b_1^* + \tau_2 \beta b_3^*}$, $T_2 = g^{\tau_1 \eta b_2^* + \tau_2 \beta b_4^*}$) if S^* is a Type I signature. If S^* is a Type II signature, $\mathcal{D}$ returns 1 and bets that $T_1 = g^{\tau_1 \eta b_1^* + \tau_2 \beta b_3^* + \tau_3 b_5^*}$, $T_2 = g^{\tau_1 \eta b_2^* + \tau_2 \beta b_4^* + \tau_3 b_6^*}$. $\square$

5.3. Claim 3. In $\text{Game}_{\text{final}}$, adversary $\mathcal{A}$ cannot output a Type I forgery signature as long as the subspace assumption holds with $k = 1$ and $l = 6$.

Proof. $\mathcal{D}$ begins by taking an instance $(\mathbb{G}, \mathbb{G}_T, p, e(\cdot, \cdot), g, g^{b_1}, g^{b_2}, g^{b_3}, g^{b_4}, g^{b_5}, g^{b_6}, g^{\eta b_1^*}, g^{\eta b_2^*}, g^{\beta b_3^*}, g^{\beta b_4^*}, g^{b_5^*}, g^{b_6^*}, U_1 = g^{\mu_1 b_1 + \mu_2 b_2 + \mu_3 b_3}, \mu_3, T_1)$. $\mathcal{D}$'s task is to decide $T_1 = g^{\tau_1 \eta b_1^* + \tau_2 \beta b_3^*}$ or $T_1 = g^{\tau_1 \eta b_1^* + \tau_2 \beta b_3^* + \tau_3 b_5^*}$.

Initialisation: $\mathcal{D}$ picks a TCR hash function $H: \{0, 1\}^* \rightarrow \mathbb{Z}_p$. The public parameter PP is $\{\mathbb{G}, \mathbb{G}_T, p, e(\cdot, \cdot), g, H\}$. $\mathcal{D}$ implicitly defines the dual orthonormal bases $\mathbb{D} = \{\mathbf{d}_1, \dots, \mathbf{d}_6\}$ and $\mathbb{D}^* = \{\mathbf{d}_1^*, \dots, \mathbf{d}_6^*\}$ as

$$\begin{aligned} \mathbf{d}_1 &= \mathbf{b}_4^*, \\ \mathbf{d}_2 &= \mathbf{b}_3^*, \\ \mathbf{d}_3 &= \mathbf{b}_5^*, \\ \mathbf{d}_4 &= \mathbf{b}_6^*, \\ \mathbf{d}_5 &= \mathbf{b}_1^*, \\ \mathbf{d}_6 &= \mathbf{b}_2^*, \\ \mathbf{d}_1^* &= \mathbf{b}_4, \\ \mathbf{d}_2^* &= \mathbf{b}_3, \\ \mathbf{d}_3^* &= \mathbf{b}_5, \\ \mathbf{d}_4^* &= \mathbf{b}_6, \\ \mathbf{d}_5^* &= \mathbf{b}_1, \\ \mathbf{d}_6^* &= \mathbf{b}_2. \end{aligned} \quad (33)$$

We note that $\mathbb{D}, \mathbb{D}^*$ are properly distributed dual orthonormal bases, and $\mathcal{D}$ can generate $g^{d_1}, g^{d_2}, g^{d_3}, g^{d_4}$ and $g^{d_1^*}, g^{d_2^*}, g^{d_3^*}, g^{d_4^*}, g^{d_5^*}$ but cannot generate g^{d_5} . $\mathcal{D}$ picks random values $\theta, \sigma, \alpha \in \mathbb{Z}_p$ and a random polynomial $F(x) \in \mathbb{Z}_p$ with degree $t-1$ and sets $F(0) = \alpha$. $\mathcal{D}$ computes

$$Z = e_\ell(g^{b_4}, g^{b_4^*})^{\alpha \theta} = e(g, g)^{\alpha \theta d_1 \cdot d_1^*}. \quad (34)$$

$\mathcal{D}$ picks random values $r_{i_1}', r_{i_2}', t_5, t_6 \in \mathbb{Z}_p$ and sets $r_{i_1} = \mu_3 r_{i_1}'$. $\mathcal{D}$ computes P_i 's partial decryption key:

$$\begin{aligned} SK_{i_1} &= (g^{b_4})^{F(i)\theta} = g^{F(i)\theta d_1^*}, \\ SK_{i_2} &= (g^{b_4})^{\mu_3 r_{i_1}'} (g^{b_5})^{r_{i_2}'} (g^{b_1})^{t_5} (g^{b_2})^{t_6} (U_1)^{-\theta r_{i_1}'} (g^{b_6})^{-r_{i_2}'} \\ &= (g^{d_1^*})^{\mu_3 r_{i_1}'} (g^{d_5^*})^{r_{i_2}'} (g^{d_5^*})^{t_5} (g^{d_6^*})^{t_6} (U_1)^{-\theta r_{i_1}'} (g^{b_6})^{-r_{i_2}'} \\ &= g^{\mu_3 r_{i_1}'} \theta d_1^* + r_{i_2} \sigma d_3^* + t_5 d_5^* + t_6 d_6^* (U_1)^{-\theta r_{i_1}'} (g^{b_6})^{-r_{i_2}'} \\ &= g^{r_{i_1} \theta d_1^* + r_{i_2} \sigma d_3^* + t_5 d_5^* + t_6 d_6^*} (U_1)^{-\theta r_{i_1}'} (g^{b_6})^{-r_{i_2}'} \\ &= g^{r_{i_1} \theta d_1^* + r_{i_2} \sigma d_3^* + t_5 d_5^* + t_6 d_6^*} (g^{\mu_1 b_1 + \mu_2 b_2 + \mu_3 b_3})^{-\theta r_{i_1}'} (g^{d_4^*})^{-r_{i_2}'} \\ &= g^{r_{i_1} \theta d_1^* + r_{i_2} \sigma d_3^* + t_5 d_5^* + t_6 d_6^*} (g^{\mu_1 d_5^* + \mu_2 d_6^* + \mu_3 d_2^*})^{-\theta r_{i_1}'} (g^{d_4^*})^{-r_{i_2}'} \\ &= g^{r_{i_1} \theta d_1^* + r_{i_2} \sigma d_3^* + t_5 d_5^* + t_6 d_6^*} g^{-\theta \mu_3 r_{i_1}'} d_2^* - \mu_1 \theta r_{i_1} d_5^* - \mu_2 \theta r_{i_1} d_6^* g^{-r_{i_2} \sigma d_4^*} \\ &= g^{r_{i_1} \theta d_1^* + r_{i_2} \sigma d_3^* + t_5 d_5^* + t_6 d_6^*} g^{-r_{i_1} \theta d_2^* - r_{i_2} \sigma d_4^* - \mu_1 \theta r_{i_1} d_5^* - \mu_2 \theta r_{i_1} d_6^*} \\ &= g^{r_{i_1} \theta d_1^* - r_{i_1} \theta d_2^* + r_{i_2} \sigma d_3^* - r_{i_2} \sigma d_4^* + t_5 d_5^* + t_6 d_6^*}, \end{aligned} \quad (35)$$

where $t_5' = t_5 - \mu_1 \theta r_{i_1}'$ and $t_6' = t_6 - \mu_2 \theta r_{i_1}'$, for $i = 1, \dots, n$.

Finally, $\mathcal{D}$ sends PP, PK = $(Z, g^{d_1}, g^{d_2}, g^{d_3}, g^{d_4})$ to adversary $\mathcal{A}$.

Query. Since $\mathcal{D}$ knows all SK_i , it can perfectly answer corruption queries and signing queries.

$\mathcal{A}$ produces a forgery signature (M^*, S^*) . Firstly, $\mathcal{D}$ must check that $\text{Verify}(\text{PK}, M^*, S^*) = \text{"true"}$; if not, then $\mathcal{D}$ will abort. In addition, $\mathcal{D}$ must verify that it has not received a Type II forgery signature. To do this, $\mathcal{D}$ can check whether $e_\ell(g^{d_5} g^{d_6}, S_1^*) = 1_{\mathbb{G}_T}$ or not. If not, the forgery signature

TABLE 1: Comparison of efficiency.

Scheme	Setup	Share-sign	Share-verify	Combine	Verify
LY13 [10]	$50(n+1)e + nE$	$3E$	$100e + E$	$2tE$	$150e + E$
LJY16 [15]	$(t^2 + 5t + 7)E$	$2lE + 18E$	$10e + 2lE$	$8(t+1)E$	$10e + 2lE$
Ours	$7e + 36E$	$13E$	$6e + 2E$	0	$6e + E$

e denotes a single bilinear pairing, E denotes a single modular exponentiation, l denotes the length of the block message, and n denotes the number of the participants.

(M^*, S^*) is a Type II signature; then $\mathcal{D}$ will abort. Notice that $\mathcal{D}$ needs the output of $\mathcal{A}$ to be a Type I forgery signature in order to defeat the subspace assumption with $k = 1$ and $l = 6$. Finally, $\mathcal{D}$ uses the forgery signature to determine what subspace T_1 is in. $\mathcal{D}$ can simply check whether $e_\ell(S_1^*, T_1) = 1_{G_T}$ or not. If so, $T_1 = g^{\tau_1 \eta b_1^* + \tau_2 \beta b_2^*}$. Otherwise, $T_1 = g^{\tau_1 \eta b_1^* + \tau_2 \beta b_2^* + \tau_3 b_3^*}$.

It comes that $\mathcal{A}$'s advantage is thus negligible if the subspace assumption with $k = 1$ and $l = 6$ holds.

According to Claims 1, 2, and 3, the signature scheme Φ is EUF-aCMA-DCA secure. $\square$

6. Comparison

Now, let us proceed to compare our scheme to other related schemes. As a reference for a further efficiency discussion, we have chosen the Libert-Yung (LY) signature scheme [10] and the second signature scheme in [15]. Let us denote the computational cost needed to perform a single bilinear pairing by e and the cost of a single modular exponentiation by E . Recall that the cost e_N of a single bilinear pairing of composite order $N = p_1 p_2 p_3$ is about 50 times more expensive than the cost e_p of a single bilinear pairing of prime order p [12]. In general, in terms of computational overhead, a single bilinear pairing is equivalent to four modular exponentiations (or $e \approx 4E$). The first column in Table 1 provides the name of signature (LY13, LJY16, and ours). Other columns detail the computational costs needed to set up signature scheme (Setup), generate shares of signature (Share-Sign), verify shares (Share-Verify), combine shares into a signature (Combine), and verify a signature (Verify).

In [15], the scheme mainly uses Groth-Sahai's [17] proof systems technique, but our scheme mainly uses dual pairing vector spaces (DPVS) technique. The main difference of these two techniques is that the DPVS technique can effectively reduce computational complexity at the cost of slightly increasing storage space.

7. Conclusion

In this paper, we have proposed an efficient (t, n) threshold signature scheme. Under the subspace assumption, we have proven that our scheme is secure against existential forgeries under adaptive chosen message attacks and adaptive corruption attacks in the standard model. Our scheme is more efficient than the LY13's and LJY16's threshold signature schemes.

Data Availability

The data supporting the findings of this study are available within the article.

Conflicts of Interest

The author declares that there are no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61972103) and Project of Enhancing School with Innovation of Guangdong Ocean University (2019-233).

References

- [1] Y. Desmedt, "Society and group oriented cryptography: a new concept," in *Advances In Cryptology—CRYPTO '87*, pp. 120–127, Springer, Berlin, Heidelberg, 1988.
- [2] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 281–308, 1988.
- [3] S. Kim, J. Kim, J. H. Cheon, and S.-H. Ju, "Threshold signature schemes for ElGamal variants," *Computer Standards & Interfaces*, vol. 33, no. 4, pp. 432–437, 2011.
- [4] V. Shoup, "Practical threshold signatures," in *Advances in Cryptology EUROCRYPT 2000*, B. Preneel, Ed., Lecture Notes in Computer Science 1807, pp. 207–220, Springer, Berlin, Heidelberg, 2000.
- [5] X.-L. Huang, Y.-X. Dong, K.-X. Jiao, and G.-D. Ye, "Asymmetric pixel confusion algorithm for images based on RSA and Arnold transform," *Frontiers of Information Technology & Electronic Engineering*, vol. 21, no. 12, pp. 1783–1794, 2020.
- [6] B. Libert and M. Yung, "Adaptively secure non-interactive threshold cryptosystems," in *The 38th International Colloquium on Automata, Languages and Programming (ICALP'2011)*, L. Aceto, M. Henzinger, and J. Sgall, Eds., Lecture Notes in Computer Science 6756, pp. 588–600, Springer, Berlin, Heidelberg, 2011.
- [7] Z. Wang, H. Qian, and Z. Li, "Adaptively secure threshold signature scheme in the standard model," *Informatica*, vol. 20, no. 4, pp. 591–612, 2009.
- [8] M. Abe and S. Fehr, "Adaptively secure feldman VSS and applications to universally-composable threshold cryptography," in *Advances in Cryptology-Crypto'2004*, M. Franklin, Ed., Lecture Notes in Computer Science 3152, pp. 317–334, Springer, Berlin, Heidelberg, 2004.
- [9] B. Waters, "Efficient identity-based encryption without random oracles," in *Advances in Cryptology C EUROCRYPT 2005*, R. Cramer, Ed., Lecture Notes in Computer Science 3494, Springer, Berlin, Heidelberg, 2005.
- [10] B. Libert and M. Yung, "Adaptively secure non-interactive threshold cryptosystems," *Theoretical Computer Science*, vol. 478, pp. 76–100, 2013.
- [11] L. Allison and B. Waters, "New techniques for dual system encryption and fully secure HIBE with short ciphertexts," in *Theory of Cryptography'2010*, D. Micciancio, Ed., Lecture

- Notes in Computer Science 5978, pp. 455–479, Springer, Berlin, Heidelberg, 2010.
- [12] D. Freeman, “Converting pairing-based cryptosystems from composite-order groups to prime-order groups,” in *Advances in Cryptology—EUROCRYPT’2010*, H. Gilbert, Ed., Lecture Notes in Computer Science 6110, pp. 44–61, Springer, Berlin, Heidelberg, 2010.
 - [13] J. Y. Hwang, H. J. Kim, D. H. Lee, and B. Song, “An enhanced (t, n) threshold directed signature scheme,” *Information Sciences*, vol. 275, pp. 284–292, 2014.
 - [14] R. Kumar, H. K. Verma, and R. Dhir, “Analysis and design of protocol for enhanced threshold proxy signature scheme based on RSA for known signers,” *Wireless Personal Communications*, vol. 80, no. 3, pp. 1281–1345, 2014.
 - [15] B. Libert, M. Joye, and M. Yung, “Born and raised distributively: fully distributed non-interactive adaptively-secure threshold signatures with short shares,” *Theoretical Computer Science*, vol. 645, pp. 1–24, 2016.
 - [16] T. P. Pedersen, “A threshold cryptosystem without a trusted party,” in *Advances in Cryptology-CRYPTO’2001*, Lecture Notes in Computer Science 547, Springer, Berlin, Heidelberg, 2001.
 - [17] J. Groth and S. Amit, “Efficient non-interactive proof systems for bilinear groups,” in *Advances in Cryptology-EUROCRYPT 2008*, N. Smart, Ed., Lecture Notes in Computer Science 4965, pp. 415–432, Springer, Berlin, Heidelberg, 2008.
 - [18] L. Harn and F. Wang, “Threshold signature scheme without using polynomial interpolation,” *International Journal of Network Security*, vol. 18, no. 4, pp. 710–717, 2016.
 - [19] H. Assidi, E. B. Ayebie, and E. M. Souidi, “An efficient code-based threshold ring signature scheme,” *Journal of Information Security and Applications*, vol. 45, pp. 52–60, 2019.
 - [20] L. Allison, “Tools for simulating features of composite order bilinear groups in the prime order setting,” in *Advances in Cryptology—Eurocrypt’2012*, D. Pointcheval and T. Johansson, Eds., Lecture Notes in Computer Science 7237, pp. 318–335, Springer, Berlin, Heidelberg, 2012.
 - [21] T. Okamoto and K. Takashima, “Homomorphic encryption and signatures from vector decomposition,” in *Pairing-Based Cryptography-Pairing’2008*, S. Galbraith and K. Paterson, Eds., Lecture Notes in Computer Science 5209, pp. 57–74, Springer, Berlin, Heidelberg, 2008.
 - [22] T. Okamoto and K. Takashima, “Some key techniques on pairing vector spaces,” in *Progress in Cryptology—Africacrypt’2011*, A. Nitaj and D. Pointcheval, Eds., Lecture Notes in Computer Science 6737, pp. 380–382, Springer, Berlin, Heidelberg, 2011.
 - [23] M. Gerbush, L. Allison, O. N. Adam, and B. Waters, “Dual form signatures: an approach for proving security from static assumptions,” in *Advances in Cryptology—ASIACRYPT’2012*, pp. 25–42, Lecture Notes in Computer Science 7658, Springer, Berlin, Heidelberg, 2012.
 - [24] D. Boneh and M. Franklin, “Identity-based encryption from the weil pairing,” in *Advances in Cryptology-Crypto’2001*, J. Kilian, Ed., Lecture Notes in Computer Science 2139, pp. 213–229, Springer, Berlin, Heidelberg, 2001.
 - [25] T. Okamoto and K. Takashima, “Adaptively attribute-hiding (hierarchical) inner product encryption,” in *Advances in Cryptology—Eurocrypt’2012*, D. Pointcheval and T. Johansson, Eds., Lecture Notes in Computer Science 7237, pp. 591–608, Springer, Berlin, Heidelberg, 2012.
 - [26] E. Kiltz, “Chosen-ciphertext secure key-encapsulation based on gap hashed Diffie-Hellman,” in *Public Key Cryptography—PKC 2007: Lecture Notes in Computer Science 4450*, T. Okamoto and X. Wang, Eds., pp. 282–297, Springer, Berlin, Heidelberg, 2007.