SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2014; **7**:2094–2103
Published online 16 December 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.921

RESEARCH ARTICLE

Simulatable and secure certificate-based threshold signature without pairings

Feng Wang[1,2], Chin-Chen Chang[2,3]* and Lein Harn[4]

[1] Department of Mathematics and Physics, Fujian University of Technology, Fuzhou, Fujian, 350108, China
[2] Department of Information Engineering and Computer Science, Feng Chia University, 100 Wenhwa Rd., Seatwen, Taichung 40724, Taiwan
[3] Department of Computer Science and Information Engineering, Asia University, Taichung 41354, Taiwan
[4] Department of Computer Science Electrical Engineering, University of Missouri–Kansas City, 5110 Rockhill Road, Kansas City, MO, 64110, U.S.A.

ABSTRACT

We propose the notion and define the security model of a certificate-based threshold signature. The model is a general model that allows both the master secret key and user secret keys to be determined and distributed to the corresponding participators. Furthermore, the model can be easily converted into an identity-based (ID-based) threshold signature model to solve the key escrow problem and can be converted into a certificateless threshold signature model. In addition, we propose a secure and efficient certificate-based threshold signature scheme. Compared with previous ID-based threshold signature and certificateless threshold signature, our scheme requires no computation of pairings and no trusted dealer. In addition, in our proposed scheme, unlike most schemes that require all members to jointly generate a certificate or a signature, it only requires t or more than t members to generate a certificate or a signature. Our proposed scheme can detect dishonest participants as well. Therefore, our scheme is more practical than existing schemes. We show that our scheme is existentially unforgeable against adaptive chosen message attacks under the discrete logarithm assumption. Copyright © 2013 John Wiley & Sons, Ltd.

KEYWORDS

certificate-based signature; threshold signature; discrete logarithm assumption; verifiable secret sharing; simulatability

***Correspondence**

Chair Professor Chin-Chen Chang, Department of Information Engineering and Computer Science, Feng Chia University, 100 Wenhwa Rd., Seatwen, Taichung 40724, Taiwan.
E-mail: alan3c@gmail.com

1. INTRODUCTION

A threshold signature scheme was first proposed by Desmedt and Frankel [1]. It is a group-oriented signature. A (t, n) threshold signature scheme can be used in the following scenario. Suppose that there is a company with n directors and the company policy requires that each document of the company must be signed by t or more than t directors. There is a single public key of the company, and the company's private key is divided into n 'shares' and gives each share to a director. With any t or more than t shares, the company's signature can be generated; however, with any $t-1$ or fewer than $t-1$ shares, the company's signature cannot be generated. This is the (t, n) threshold signature scheme.

Obviously, the (t, n) threshold signature scheme is very similar to the threshold secret-sharing scheme [2] or the distributed key generation (DKG) protocol [3,4]. The main difference between them is that the secret will be exposed after reconstruction in the latter schemes, but not in the threshold signature scheme. However, we can use either the threshold secret-sharing scheme or the DKG protocol to construct a (t, n) threshold signature scheme.

In traditional public-key cryptography (PKC), a user's public key is required to be certified by a certification authority (CA). This procedure requires complicated certificate management. To solve this problem, Shamir [5] proposed the ID-based PKC (ID-PKC). In ID-PKC, the user's public key is his or her identity such as an email address, and his or her private key is generated by the private key generator (PKG). Therefore, PKG knows a user's private key, which is known as the private key escrow problem [6]. Although ID-PKC has simplified the certificate management problem, it creates a key escrow problem.

There are two approaches to solve the key escrow problem. One is the certificateless PKC (CL-PKC) [7],

which was proposed in 2003. In that approach, the user's private key is composed of a partial private key and a secret key. The partial private key is generated from the user's identity by the key generation center. The secret key is chosen by the user, and a corresponding public key is published without certificate. Another approach is the certificate-based PKC (CB-PKC) [8], which was proposed in 2003. In that approach, each user selects his or her secret key and generates a corresponding public key. Then, it requests a certificate from a CA. The user combines both his or her secret key and the certificate to form his or her private key.

Furthermore, most schemes of the ID-PKC, CL-PKC and CB-PKC are constructed using bilinear pairings, which require more computational cost than normal operations in $GF(p)$ or Z_n, where $GF(p)$ denotes Galois field with prime p elements and Z_n denotes the set of modulo positive integer n. Therefore, there is an interesting research topic in constructing secure schemes of the ID-PKC, CL-PKC and CB-PKC without using pairings. In 2013, Li *et al*. [9] proposed a provably secure certificate-based signature (CBS) scheme using the discrete logarithm (DL) assumption.

In general, there are four types of threshold signatures, which can be described as follows. The threshold signature in traditional PKC has been studied in [1,10,11]. Desmedt and Frankel [1] proposed the first (t, n) threshold signature scheme based on the Rivest–Shamir–Adleman assumption and Shamir's secret sharing. Harn [10] combined a modified ElGamal scheme and Shamir's secret sharing into a (t, n) threshold signature scheme. Kim *et al*. [11] extended Harn's scheme to all ElGamal variants.

Combined with the ID-PKC, Beak and Zhang [12] proposed the first ID-based threshold signature (IDTS) in 2004. In that scheme, the PKG generates the user's private key and distributes its shares to the signature generation servers (SGSs). Chen *et al*. [13] proposed another IDTS without a trusted PKG. This scheme is a certificateless threshold signature (CLTS) scheme, but its security analysis is made in the model of IDTS.

Combined with the CL-PKC, Wang *et al*. [14] proposed the first CLTS scheme in 2007. Their CLTS model is a general model in which both master secret key and user private key are shared among corresponding parties; however, their scheme requires a PKG clerk to distribute the master key and user partial private key using a secret-sharing scheme. This implies that the PKG clerk knows all partial private keys and the master key in the CL-PKC. So, the PKG clerk does nothing but reduces its efficiency. Furthermore, Yuan *et al*. [15] pointed out that this scheme cannot detect any misbehavior of dishonest participants, and they have proposed a new CLTS scheme. In the scheme of Yuan *et al*., the CLTS model is not general because the master key is known by the PKG only, and the efficiency is poor because their signature scheme requires that all the SGSs attend in the signature generation phase.

Combined with CB-PKC, there is a certificate-based threshold encryption [16]. As for the certificate-based threshold signature (CBTS), we have not seen any related studies in the literature.

Inspired by the findings mentioned earlier, in this paper, we define the formal notion of a general CBTS and its security model and extend the CBS scheme of Li *et al*. into a CBTS scheme, which is secure under the DL assumption. In the following, we summarize our contributions.

- We present a formal notion of CBTS and its security model. The proposed CBTS model is a general model.
- Our CBTS scheme is based on the DL assumption without pairings.
- Our CBTS model can be easily converted into a CLTS model, and our scheme can solve the problem in both the schemes of Wang *et al*. and Yuan *et al*. mentioned earlier.
- Our CBTS model can be easily converted into an IDTS model to solve the key escrow problem.

The rest of the paper is organized as follows. Section 2 gives the definition and security model of CBTS and discusses the relationship between CBTS and the underlying CBS using the concept of a simulatable view. Section 3 introduces a DL assumption and some building blocks for our scheme, such as a verifiable secret-sharing scheme and the CBS scheme in [9]. We propose our scheme in Section 4 and give a formal security proof in Section 5. In Section 6, we compare our scheme with others, and the conclusion is given in Section 7.

2. CERTIFICATE-BASED THRESHOLD SIGNATURES

Inspired by the model of threshold signature in [14,15], we present a formal notion of CBTS and its security model. Furthermore, we discuss the relationship between the CBTS and the underlying CBS using the concept of a simulatable view.

2.1. Definition of certificate-based threshold signature

In a CBTS, there are four entities: a CA, a (n_C, t_C) certificate generation server (CGS), a (n_S, t_S) SGS and a verifier. A CA generates the system public parameters, *params*. The n_C CGSs collectively generate the master secret key, *msk*, master public key, *mpk*, and certificate, *Cer*. In this procedure, each CGS_i has a share, msk_i, of *msk*, and any t_C or more than t_C CGSs can cooperate to generate certificate *Cer* for the *upk* (user public key) using their share, msk_i; however, any $t_C - 1$ or fewer than $t_C - 1$ CGSs cannot. We say t_C is the threshold parameter and denote this entity as (n_C, t_C) CGSs. Similarly, the n_S SGSs collectively generate the user secret key, *usk*, user public key, *upk*, and signature, σ. We denote this entity as

(n_S,t_S) SGSs, where t_S is the threshold parameter. A verifier checks whether the signature is valid or not.

Definition 1(CBTS). Formally, a CBTS scheme involves six algorithms.

Setup: Given a security parameter $k \in N$, CA generates its system public parameters, *params*.

MasterKeyGen: Given *params*, n_C CGSs collectively generate a concealed master secret key, *msk*, and a master public key, *mpk*. Each CGS_i has a share msk_i of *msk*.

UserKeyGen: Given *params* and user identity, *ID*, the n_S SGSs cooperate together to generate the concealed user secret key, *usk*, and the user public key, *upk*. Each SGS_i has a share usk_i of *usk*.

Certify: Given *params*, *mpk*, *ID* and *upk*, any t_C or more than t_C CGSs with their share, msk_i, collectively generate user certificate, *Cer*, and distribute the share, Cer_i, of *Cer* to n_S SGSs via a secure channel.

Sign: Given *params*, *mpk*, *ID*, *upk* and message, *m*, any t_S or more than t_S SGSs with their secret key share, usk_i, and certificate share, Cer_i, collectively generate a signature, σ, for *m*.

Verify: Given *params*, *mpk*, *ID*, *PK*, *m* and σ, the verifier checks whether the σ is valid or not.

2.2. Security model of certificate-based threshold signatures

According to the security model of a CBS described in [9] and that of CLTS described in [14,15], we consider two types of adversaries in a CBTS: $\mathcal{B}_I$ and $\mathcal{B}_{II}$. $\mathcal{B}_I$ simulates a malicious user who replaced the public key with a value of his or her choice but does not know the master key. $\mathcal{B}_{II}$ simulates a malicious certifier who knows the master key but he or she does not replace the public key.

Furthermore, we assume that each type of adversary can corrupt up to $t_C - 1$ certificate generation servers, denoted by $CGS_1, CGS_2, \ldots, CGS_{t_C-1}$ without losing the generality, and $t_S - 1$ SGSs, denoted by $SGS_1, SGS_2, \ldots, SGS_{t_S-1}$. The corrupted parties are chosen at the beginning of the protocol by the adversary. For describing the security of the CBTS scheme, we define two games of interaction between the challenger *C* and adversary $\mathcal{B}_I(\mathcal{B}_{II})$ as follows. We illustrate the interaction between the Challenger and the Adversaries in the Figure 1.

Game 1. This game is performed between a challenger *C* and the adversary $\mathcal{B}_I$ of CBTS.

Phase 1: Challenger *C* runs the algorithm, Setup, of CBTS, and returns system public parameter, *params*, to $\mathcal{B}_I$.

Phase 2: $\mathcal{B}_I$ corrupts $t_S - 1$ SGSs and $t_C - 1$ certificate generation servers.

Phase 3: Challenger *C* runs the algorithm, MasterKeyGen, of CBTS, returns master public key, *mpk*, to $\mathcal{B}_I$. Note that the corrupted shares of *msk* are available to $\mathcal{B}_I$.

Phase 4: $\mathcal{B}_I$can adaptively make the following queries to *C*:

UserKeyGen Query: *C* maintains a list, *L*, which is initially null and is used to record the information interacted with $\mathcal{B}_I$. On a query of a user's identity, *ID*, if the *ID* is already in the list, *L*, *C* returns the *upk* to $\mathcal{B}_I$. Otherwise, *C* runs the algorithm, UserKeyGen, of CBTS to obtain the user secret key share, usk_i, of *usk* and user public key, *upk*, and returns the *upk* to $\mathcal{B}_I$. *C* adds $(ID, upk, usk, usk_1, usk_2, \ldots, usk_{n_C})$ to the list *L*.

UPKReplace Query: On a query of a user's identity, *ID*, and user public key, *upk*′, *C* replaces the user's original public key, *upk*, with *upk*′.

UserSecretKey Query: On a query of a user's identity, *ID*, *C* checks if *ID* is in the list *L*. If so, *C* returns the corresponding shares of the user secret key, *usk*, but otherwise returns nothing.

Certification Query: On a query of a user's identity, *ID*, *C* runs the algorithm Certify to obtain the user's certificate share, Cer_i, of *Cer* and returns the *Cer* to $\mathcal{B}_I$.

Sign Query: On query of a user's identity, *ID*, and message, *m*, *C* runs the algorithm Sign of CBTS to obtain the signature σ and returns the σ to $\mathcal{B}_I$.

Phase 5: $\mathcal{B}_I$ submits the target ID^* to *C*. *C* runs the algorithms, UserSecretKey Query and Certification Query, to obtain the user secret key share, usk_i, of *usk* and the user's certificate share, Cer_i, of *Cer*. The corrupted shares of *usk* and *Cer* are available to $\mathcal{B}_I$.

Phase 6: Finally, $\mathcal{B}_I$ outputs a signature, σ^*, of the message, m^*, under the identity, ID^*, and the corresponding public key, PK^*. We say that $\mathcal{B}_I$ wins the game if σ^* is a valid signature, and (ID^*, m^*, PK^*) has not been requested to Sign Query.

We denote the successful probability that $\mathcal{B}_I$ wins in the aforementioned game by $Succ_{B_I}$.

Game 2. This game is performed between a challenger *C* and adversary $\mathcal{B}_{II}$ for CBTS. The game is the same as game 1 except in phases 3 and 4 as follows.

DOI: 10.1002/sec

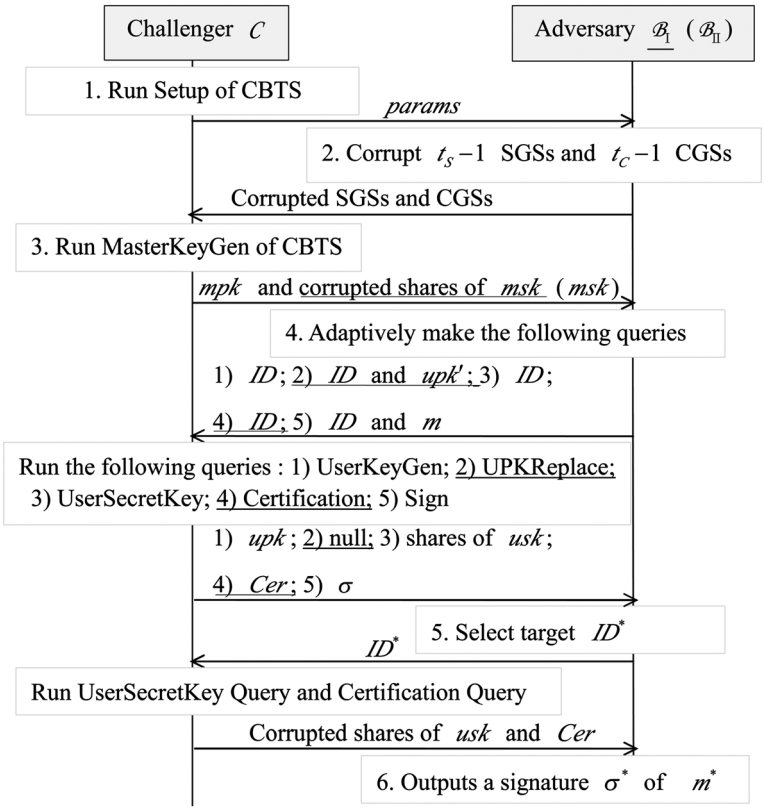

Figure 1. The games of interaction between challenger and adversary of CBTS. *Note: The content of this figure without the inside of the parentheses depicts game 1, which is performed between challenger C and adversary $\mathcal{B}_I$. The content that replaces the underlined part into the inside of the parentheses depicts game 2, which is performed between challenger C and adversary $\mathcal{B}_{II}$.*

Phase 3: Challenger *C* runs the algorithm, MasterKeyGen, and returns the master public key, *mpk*, and master secret key, *msk*, to $\mathcal{B}_{II}$.

Phase 4: $\mathcal{B}_{II}$ can adaptively make a UserKeyGen Query, UserSecretKey Query and Sign Query as described in Game 1.

We denote the successful probability that $\mathcal{B}_{II}$ wins in the preceding game by $Succ_{B_{II}}$.

Definition 2. We say that a CBTS scheme is existentially unforgeable against adaptively chosen message attacks (EUF-CBTS-CMA) if the successful probablity of any polynomially bounded adversary in the earlier two games is negligible. Similarly, we denote that a CBS scheme is existentially unforgeable against adaptively chosen message attacks by EUF-CBS-CMA.

2.3. Relationship between EUF-CBTS-CMA and EUF-CBS-CMA

Motivated by [3,14,15], we use the concept of a simulatable view to prove the unforgeability of a CBTS scheme. The simulatability of a scheme means that the scheme is simulatable by a simulator. On input of the public value and all corrupted information, the simulator can output a distribution that is computationally indistinguishable from the view of an adversary that interacts with honest parties in a regular run of a protocol that ends with the public values as its public output. This is to say that the corrupted information does not provide any useful information to the adversary other than the public information in the protocol. We extend the notion of simulatability to CBTS and describe the relationship between EUF-CBTS-CMA and EUF-CBS-CMA as follows.

Definition 3. (Simulatability of CBTS). The simulatability of CBTS scheme means that its algorithms, MasterKeyGen, UserKeyGen, Certify and Sign, are all simulatable. We denote the corresponding simulator as *SIM–msk*, *SIM–usk*, *SIM–Cer* and *SIM–sign*, respectively.

Theorem 1. If the CBTS scheme is simulatable and the underlying CBS scheme is EUF-CBS-CMA, then the CBTS is EUF-CBTS-CMA.

Proof. In order to prove the theorem, we will show that if an adversary $\mathcal{B}_I$(or $\mathcal{B}_{II}$) can break CBTS, then there will inevitably be an adversary $\mathcal{A}_I$(or $\mathcal{A}_{II}$) [9] that can break

the underlying CBS. We denote them by type 1 and type 2 interaction, respectively.

We first describe a type 1 interaction. Let CBTS be simulatable, then we will show that how the view of $\mathcal{B}_{\mathrm{I}}$ in the real attack of Game 1, which we denote by $G_{B_{\mathrm{I}}}$, can be simulated to obtain a new game $G_{A_{\mathrm{I}}}$ [9], which is associated with the adversary $\mathcal{A}_{\mathrm{I}}$ in CBS.

Let $E_{B_{\mathrm{I}}}$ denote the event that $\mathcal{B}_{\mathrm{I}}$ outputs a valid forgery signature and $E_{A_{\mathrm{I}}}$ denote the event that $\mathcal{A}_{\mathrm{I}}$ outputs a valid forgery signature.

In order to achieve our proof, we consider $\mathcal{A}_{\mathrm{I}}$ as the challenger of $\mathcal{B}_{\mathrm{I}}$. In the following interaction, we describe the difference from Game 1 only.

In phase 3 of game 1, $\mathcal{A}_{\mathrm{I}}$ runs *SIM–msk* to obtain the shares of *msk* and sends $t_{\mathrm{C}}-1$ shares of *msk* to $\mathcal{B}_{\mathrm{I}}$. Note that neither $\mathcal{B}_{\mathrm{I}}$ nor $\mathcal{A}_{\mathrm{I}}$ knows the master secret key, *msk*.

In phase 4 of game 1, if $\mathcal{B}_{\mathrm{I}}$ wants to request the UserKeyGen Query or UPKReplace Query of *ID* to his challenger, *C*, in CBTS, he or she sends the query to $\mathcal{A}_{\mathrm{I}}$. $\mathcal{A}_{\mathrm{I}}$ sends the query to his or her challenger in CBS and obtains the corresponding value, and then $\mathcal{A}_{\mathrm{I}}$ returns them to $\mathcal{B}_{\mathrm{I}}$.

If $\mathcal{B}_{\mathrm{I}}$ wants to request the UserSecretKey Query, Certification Query or Sign Query of *ID* to his or her challenger, *C*, in CBTS, he or she sends the query to $\mathcal{A}_{\mathrm{I}}$. $\mathcal{A}_{\mathrm{I}}$ sends the query to his or her challenger in CBS and obtains the corresponding values. Then, $\mathcal{A}_{\mathrm{I}}$ runs the associated simulator defined in Definition to obtain the shares of the values and returns them to $\mathcal{B}_{\mathrm{I}}$.

In phase 5 of game 1, if $\mathcal{B}_{\mathrm{I}}$ wants to submit the target ID^* to his or her challenger, *C*, in CBTS for UserSecretKey Query and Certify Query, he or she sends the query to $\mathcal{A}_{\mathrm{I}}$. $\mathcal{A}_{\mathrm{I}}$ runs the associated simulator defined in Definition to obtain the shares of the values and returns $t_{\mathrm{S}}-1$ of them to $\mathcal{B}_{\mathrm{I}}$.

In phase 6 of game 1, when $\mathcal{B}_{\mathrm{I}}$ outputs a valid signature forgery, then $\mathcal{A}_{\mathrm{I}}$ sets this forgery as his or her own forgery.

Hence, we have $pr[E_{B_{\mathrm{I}}}] \le pr[E_{A_{\mathrm{I}}}]$, where $pr[E_{B_{\mathrm{I}}}]$ denotes the probability of $E_{B_{\mathrm{I}}}$ and $pr[E_{A_{\mathrm{I}}}]$ denotes the probability of $E_{A_{\mathrm{I}}}$.

As for the type 2 interaction, it is similar to the type 1 interaction except for the following.

In phase 3 of game 2, $\mathcal{A}_{\mathrm{II}}$ runs *SIM–msk* to obtain the shares of *msk* and sends all of the shares to $\mathcal{B}_{\mathrm{II}}$. Note that both $\mathcal{B}_{\mathrm{II}}$ and $\mathcal{A}_{\mathrm{II}}$ know the master key, *msk*.

In phase 4 of game 2, $\mathcal{B}_{\mathrm{II}}$ needs to interact $\mathcal{A}_{\mathrm{II}}$ with the UserKeyGen Query, UserSecretKey Query and Sign Query only.

In phase 5 of game 2, $\mathcal{B}_{\mathrm{II}}$ needs to interact $\mathcal{A}_{\mathrm{II}}$ with the UserSecretKey Query only.

3. BUILDING BLOCKS

In this section, we simply review some blocks for the construction of our CBTS scheme in Section 4. These blocks are the DL assumption, CBS scheme of Li *et al*. [9] and verifiable secret-sharing scheme [4,17,18].

3.1. Discrete logarithm assumption

Definition 4. (DL assumption). Given a large prime number pair (p, q), which satisfies $q|p-1$, G is a subgroup of Z_p^* with order q. g is a generator of G and elements $y \in G$. The DL problem in G is to output $\alpha \in Z_q^*$ such that $y = g^{\alpha}$.

We say that the (ε, t)-DL assumption holds in a group G if no algorithm running in time at most t can solve the DL problem in G with an advantage of at least ε.

3.2. CBS scheme of Li *et al*

In 2013, Li *et al*. [9] proposed a secure CBS scheme based on the DL assumption in the random oracle model. We simply review the scheme as follows.

Scheme 1. (CBS scheme of Li *et al*.).

Setup: Given a security parameter 1^k, where $k \in N$, CA works as follows. CA generates two primes, p and q, such that $q|p-1$, selects a generator, $g \in Z_p^*$, and chooses three cryptographic hash functions, $H_0 : \{0,1\}^* \times Z_p^* \times Z_p^* \rightarrow Z_q^*$, $H_1 : \{0,1\}^* \times Z_p^* \times Z_p^* \times Z_p^* \rightarrow Z_q^*$ and $H_2 : \{0,1\}^* \times Z_p^* \times Z_p^* \times Z_p^* \times Z_p^* \rightarrow Z_q^*$. CA publishes the system parameters as $params = < p, q, g, y, H_0, H_1, H_2 >$.

MasterKeyGen: Given *params*, CA picks a random number $msk \in Z_q^*$ as master secret key and computes $mpk = g^{msk} \bmod p$ as the master public key.

UserKeyGen: Given *params*, the user selects a random number, $usk \in Z_q^*$, as his or her user secret key and computes $upk = g^{usk} \bmod p$ as his or her user public key.

Certify: Given *params*, *mpk*, *msk*, *upk* and user identity, $ID \in \{0,1\}^*$, CA randomly picks $s \in Z_q^*$ and computes $W = g^s \bmod p$, $R = s + msk \cdot H_0(ID, mpk, W) \bmod q$, and outputs the user's certificate, $Cer = < W, R >$. Then, CA sends *Cer* to the user via a secure channel.

Sign: Given *params*, *mpk*, *ID*, *upk*, *usk*, *Cer* and message, $m \in \{0,1\}^*$, the user works as follows. The user chooses a random number, $r \in Z_q^*$, and computes $U = g^r \bmod p$, $h_1 = H_1(m, upk, U, W)$, $h_2 = H_2(m, ID, upk, U, W)$ and $z = R + usk \cdot H_1 + r \cdot H_2 \bmod q$. The signature is $\sigma = < U, W, z >$.

Verify: Given *params*, *mpk*, *ID*, *upk*, *m* and σ, the verifier computes $h_0 = H_0(ID,upk, W)$, $h_1 = H_1(m,upk,U,W)$ and $h_2 = H_2(m,ID,upk,U,W)$ and checks equation $g^z = W \cdot mpk^{h_0} \cdot upk^{h_1} \cdot U^{h_2} \mod p$. If the equality holds, the verifier accepts the signature; otherwise, he or she rejects it.

Theorem 2. Under the DL assumption, the Li *et al*. CBS scheme described in Scheme 1 is existentially unforgeable against adaptive chosen message and identity attacks in the random oracle model.

3.3. Verifiable secret sharing

In order to extend the Li *et al*. CBS scheme to a CBTS scheme, we need to share the master secret key, *msk*, among CGSs and share the user secret key, *usk*, among SGSs. This can be achieved by using a (*t*, *n*)-secret-sharing scheme based on the DL assumption. Therefore, we will briefly review the (*t*, *n*)-secret-sharing scheme described in [4,17,18]. A secret-sharing scheme means that a dealer wants to share a secret, $s \in Z_q^*$, among *n* parties, $P_1, P_2, \ldots, P_n$, such that any *t* or more than *t* parties can easily reconstruct the secret *s*, but not $t-1$ or less than $t-1$ parties. In the following scheme, *p*, *q* and *g* are the same parameters as in Definition .

Scheme 2. (Feldman's verifiable secret sharing (Feldman-VSS) [17]).

A dealer randomly selects a polynomial $f(z) = \sum_{k=0}^{t-1} a_k \cdot z^k$ over Z_q such that $f(0) = s$. He or she then sends the share $s_i = f(i) \mod q$ to P_i, for $i = 1, 2, \ldots, n$, via a secure channel and broadcasts the verfication values, $A_k = g^{a_k} \mod p$, for $k = 0, 1, \ldots, t-1$. If a party P_i finds that his or her share, s_i, does not satisfy the equation, $g^{s_i} = \prod_{k=0}^{t-1} (A_k)^{i^k} \mod p$ (equation 1), then he broadcasts a complaint against the dealer. The dealer reveals the share, s_i. If the share satisfies equation 1, P_i is disqualified; otherwise, the dealer is disqualified. At the reconstruction time, equation 1 is also used to detect any dishonest parties.

Scheme 3. (Pedersen's verifiable secret sharing (Pedersen-VSS) [18]).

This scheme is similar to Feldman-VSS in Scheme 2 except the following modifications. The dealer selects two random polynomials, one is the same as the one in Feldman-VSS and the other is $f'(z) = \sum_{k=0}^{t-1} b_k \cdot z^k$ over Z_q. Furthermore, the share changes into (s_i, s'_i), where $s'_i = f'(i) \mod q$; the verfication values change into $C_k = g^{a_k} h^{b_k} \mod p$, where *h* is in the subgroup of Z_p^* generated by *g* and $\log_g h$ is unknown. Equation 1 changes into $g^{s_i} h^{s'_i} = \prod_{k=0}^{t-1} (C_k)^{i^k} \mod p$ (equation 2).

Scheme 4. (Joint Pedersen-VSS DKG (JP-DKG) protocol [4]).
Generating *s*.

Step 1. Each party, P_i, performs a Pedersen-VSS of a random value, z_i, as a dealer. Therefore, each party, P_i, has $C_{ik} = g^{a_{ik}} h^{b_{ik}} \mod p$ for $k = 0, 1, \ldots, t-1$, and $s_{ij} = f_i(j) \mod q$, $s'_{ij} = f'_i(j) \mod q$ for $j = 1, 2, \ldots, n$.

Step 2. Each party, P_j, builds the set of non-disqualified parties, *QUAL*, by checking if $g^{s_{ij}} h^{s'_{ij}} = \prod_{k=0}^{t-1} (C_{ik})^{j^k} \mod p$ (equation 3) holds.

Step 3. Each party, P_i, sets his or her share of secret as $x_i = \sum_{j \in QUAL} s_{ji} \mod q$, and the value, $x'_i = \sum_{j \in QUAL} s'_{ji} \mod q$. The distributed secret value is a concealed $s = \sum_{i \in QUAL} z_i \mod q$.

Extracting $y = g^s \mod p$.

Step 4. Each party $i \in QUAL$ exposes $y_i = g^{z_i} \mod p$ via Feldman-VSS.

Step 5. If the value of party, P_i, does not satisfy the equation $g^{s_{ij}} = \prod_{k=0}^{t-1} (A_{ik})^{j^k} \mod p$ (equation 4), the other parties run the reconstruction phase of Pedersen-VSS to compute $z_i, f_i(z), A_{ik}$ for $k = 0, 1, \ldots, t-1$ in the clear. For all parties in *QUAL*, set $y_i = A_{i0} = g^{z_i} \mod p$ and compute $y = \prod_{i \in QUAL} y_i \mod p$.

4. OUR CBTS SCHEME

With the building blocks described in Section 3, we propose a CBTS scheme. Our scheme has six phases, including Setup, SystemKeyGen, UserKeyGen, Certify and Sign phase. We show the functions and relationships among them in Figure 2 and give the algorithm in detail as follows.

Scheme 5. (Our CBTS scheme).

Setup: Identical to the algorithm Setup of the Li *et al*. CBS scheme in Scheme 1.

SystemKeyGen: Given *params*, the n_C CGSs perform an instance of the JP-DKG protocol. Each $CGS_i \in QUAL_C$ holds an additive share, (msk_i, msk'_i), of master secret key, *msk*, whereas master public key, $mpk = g^{msk} \mod p$, and every CGS_i's $mpk_i = g^{msk_i} \mod p$ are public. Each value, msk_i, is CGS_i's secret value share with Feldman-VSS and Pedersen-VSS.

UserKeyGen: Given *params* and user identity, $ID \in \{0,1\}^*$, the n_S SGSs perform an instance of the JP-DKG protocol. Each $SGS_i \in QUAL_S$ holds an additive share, (usk_i, usk'_i), of the user secret key, *usk*, whereas the user public key, $upk = g^{usk} \mod p$, and every SGS_i's $upk_i = g^{usk_i} \mod p$ are public. Each value, usk_i, is SGS_i's secret value share with Feldman-VSS and Pedersen-VSS.

Certify:

Step 1. Given *params*, *mpk*, *upk* and *ID*, the t_C or more than t_C CGSs of $QUAL_C$ perform an instance of the JP-DKG protocol. Each $CGS_i \in QUAL'_C$

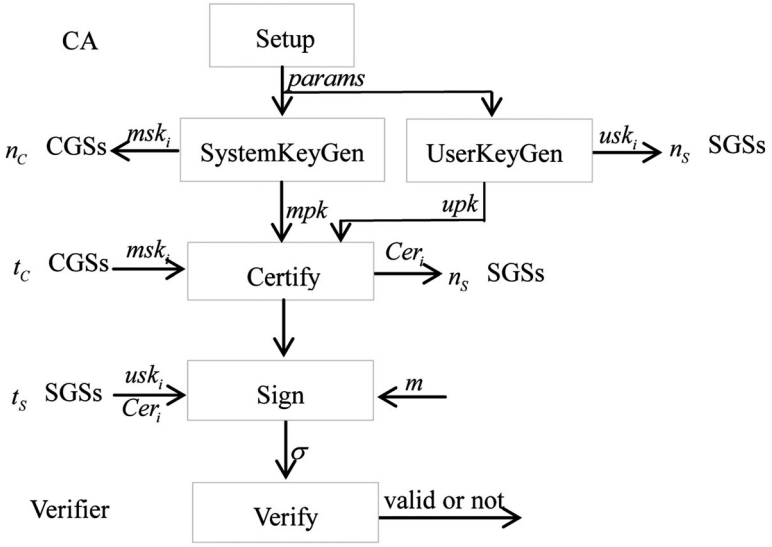

Figure 2. The flowchart of our proposed CBTS scheme.

($|QUAL'_C| \geq t_C$) holds an additive share, (s_i,s'_i), of the secret value, s. Each value s_i is CGS_i's secret value share with Feldman-VSS and Pedersen-VSS. We denote the generated public values, $W = g^s \bmod p$ and $W_i = g^{s_i} \bmod p$, for every CGS_i. Each $CGS_i \in QUAL'_C$ locally computes $h_0 = H_0(ID,upk,W)$ and broadcasts its additive share $R_i = s_i + h_0 \cdot msk_i \bmod q$. For $CGS_i \in QUAL_C - QUAL'_C$, all $CGS_i \in QUAL'_C$ run the reconstruction phase of Pedersen-VSS to compute msk_i and set $R_i = h_0 \cdot msk_i \bmod q$. The certificate is ($W$, R), where $R = \sum_{i \in QUAL_C} R_i$.

Step 2. Denoted as $R' = \sum_{CGS_i \in QUAL_C - QUAL'_C} R_i$, the $CGS_i \in QUAL'_C$ randomly pick $R'_j \in Z_q^*$, for $j = 1, 2, \ldots, |QUAL_S - QUAL'_C|$, satisfying $\sum_j R'_j = R'$. They run JP-DKG with the value, R'_j, and every CGS_i's secret value, R_i, as their own random selected secret value. Then, they send the corresponding value and W to $SGS_i \in QUAL_S$ via a public or secret channel. The method of sending must ensure that the knowledge of each $SGS_i \in QUAL_S$ is actually the same as when they run JP-DKG themselves. Therefore, each $SGS_i \in QUAL_S$ holds an additive share, (R_i,R'_i), of user partial certificate, R, whereas the value, W, and every SGS_i's $W_i = g^{R_i} \bmod p$ are public. Each value, R_i, is SGS_i's secret value shared with Feldman-VSS and Pedersen-VSS.

Sign: Given *params*, *mpk*, *ID*, *upk* and message $m \in \{0,1\}^*$, the t_S or more than t_S SGSs of $QUAL_S$ perform an instance of JP-DKG protocol. Each $SGS_i \in QUAL'_S$ ($|QUAL'_S| \geq t_S$) holds an additive share (r_i,r'_i) of secret value, r, each value, r_i, is SGS_i's secret value share with Feldman-VSS and Pedersen-VSS. We denote the generated public values, $U = g^r \bmod p$ and $U_i = g^{r_i} \bmod p$, for every SGS_i. Each $SGS_i \in QUAL'_S$ with values, usk_i and R_i, locally computes $h_1 = H_1(m,upk,U,W)$ and $h_2 = H_2(m,ID,upk,U,W)$ and broadcasts its additive share, $z_i = R_i + usk_i \cdot h_1 + r_i \cdot h_2 \bmod q$. For $SGS_i \in QUAL_S - QUAL'_S$, all $SGS_i \in QUAL'_S$ run the reconstruction phase of Pedersen-VSS to compute usk_i, R_i, and set $z_i = R_i + usk_i \cdot h_1 \bmod q$. The signature is $\sigma = <U, W, z>$, where $z = \sum_{i \in QUAL_S} z_i$.

Verify: This identical to the algorithm Verify of the Li *et al*. CBS scheme in Scheme 1.

5. SECURITY ANALYSIS OF OUR CBTS SCHEME

According to Theorem , in order to prove EUF-CBTS-CMA of our CBTS scheme, we only need to show that the underlying scheme (i.e. the Li *et al*. CBS scheme) is EUF-CBS-CMA and our scheme is simulatable. The EUF-CBS-CMA of the Li *et al*. CBS scheme has been proven in [9] as described in Theorem in Section 3. Thus, we only need to prove the simulatability of our CBTS scheme.

Theorem 3. Our CBTS scheme is simulatable.

Proof. We describe four simulators, SIM–msk, *SIM–usk*, *SIM–Cer* and *SIM–sign*, of our CBTS scheme to ensure its simulatablity as follows.

The simulators *SIM–msk* and *SIM–usk* can be constructed in the same way as in Figure 3 of [4]. We simply refer to

them as simulator 1. The simulators *SIM–Cer* and *SIM–sign* are referred to as simulators 2 and 3, respectively. In light of Definition , we can prove our theorem.

Simulator 1. (The simulator of JP-DKG (*SIM–JPDKG*)). We denote the set of corrupted parties controlled by an adversary as $A=\{1,2,\ldots,t'-1\}$ and the parties controlled by the simulator as $sim=\{t',t'+1,\ldots,n\}$, where $t'\le t$. With input public key, y, the simulator performs as follows.

Step 1. The simulator performs step 1–3 in JP-DKG on behalf of P_i, for $i\in sim$. The adversary view consists of $f_i(z)$, $f'_i(z)$, s_{ij}, s'_{ij}, C_{ik}, for $i\in A$, $j\in QUAL$, and $k=0,1,\ldots,t-1$. The simulator knows all $f_i(z)$, $f'_i(z)$, s_{ij}, s'_{ij}, C_{ik},for $i,j\in QUAL$, $k=0,1,\ldots,t-1$.

Step 2. The simulator performs the algorithm similar to steps 4 and 5 in JP-DKG except that A_{nk} for $k=0,1,\ldots,t-1$. In this process, the simulator computes $A_{n0}=y\cdot\prod_{i\in\{QUAL/\{n\}\}}A_{i0}^{-1}$ and $A_{nk}=A_{n0}^{\lambda_{k0}}\cdot\prod_{i=1}^{t-1}(g^{s_{ni}})^{\lambda_{ki}}$, for $k=1,\ldots,t-1$, where λ_{ki}'s are the Lagrange interpolation coefficients.

Simulator 2. (The simulator of certify (*SIM–Cer*)).

We denote the set of corrupted CGSs controlled by the adversary as $A_C=\{1,2,\ldots,t'_C-1\}$ and the CGSs controlled by the simulator as $sim_C=\{t'_C,t'_C+1,\ldots,n_C\}$, where $t'_C\le t_C$. Given *params*, *mpk*, *ID*, *upk*, *W* and msk_i, R_i, W_i, where $i\in A_C$, the simulator performs as follows.

Step 1. The simulator runs *SIM–msk* with input *mpk*, on behalf of CGS_i for $i\in sim_S-\{n_S\}$. At the end of the simulation, it outputs a probability distribution identical to the one produced in a regular run of UserKeyGen in Scheme 5. We notice that mpk_{n_C} (Denoted by A_{n0} in *SIM – JPDSG*, which is similar to *SIM – msk*) is different from the value generated in UserKeyGen. Then, the simulator picks two random values, $h_0,R_{nc}\in Z_q^*$, and computes $W_{n_C}=g^{R_{n_C}}\cdot mpk_{n_C}^{-h_0}$. The simulator sets $H_0(ID,upk,W)=h_0$, where we regard H_0 as random oracle machine.

Step 2. The simulator runs *SIM–JPDKG* with input W/W_{n_C} on behalf of CGS_i for $i\in sim_C-\{n_C\}$. As for CGS_{n_C}, it simulates Pedersen-VSS and broadcasts W_{n_C}. Then, the simulator performs algorithm step 1 of Certify in Scheme 5 on behalf of $CGS_1,CGS_2,\ldots,CGS_{n_C-1}$. It is a valid certificate because $g^R=g^{\sum R_i}=\prod g^{R_i}=\prod\left(W_i\cdot mpk_i^{h_0}\right)=W\cdot mpk^{h_0}$.

Step 3. The simulator runs algorithm step 2 of Certify in Scheme 5 similarly and sends the values to $SGS_i\in QUAL_S$.

Simulator 3. (The simulator of sign (*SIM–sign*)).

We denote the set of corrupted SGSs controlled by the adversary as $A_S=\{1,2,\ldots,t'_s-1\}$, and the SGSs controlled by simulator as $sim_S=\{t'_s,t'_s+1,\ldots,n_s\}$, where $t'_s\le t_s$. Given *params*, *mpk*, *ID*, *upk*, *m*, *U*, *W*, and usk_i, upk_i, R_i, W_i, U_i, where $i\in A_S$, the simulator performs as follows.

Step 1. This step is similar to Step 1 of *SIM–Cer* in simulator 2. The simulator runs *SIM–usk* with input *upk*, *SIM–JPDKG* with input *W* on behalf of SGS_i for $i\in sim_S$. Then, the simulator picks three random values $h_1,h'_2,z_{n_S}\in Z_q^*$, computes $U_{n_S}=\left(g^{z_{n_S}}\cdot W_{n_S}^{-1}\cdot upk_{n_S}^{-h_1}\right)^{h'_2}$ and sets $H_1(m,upk,U,W)=h_1$ and $H_2(m,ID,upk,U,W)=(h'_2)^{-1}\bmod q$.

Step 2. This step is similar to Step 2 of *SIM–Cer* in simulator 2. The simulator runs *SIM–JPDKG* with input U/U_{n_S} on behalf of CGS_i for $i\in sim_S-\{n_S\}$. As for CGS_{n_S}, it simulates Pedersen-VSS and broadcasts U_{n_S}. Then, it performs algorithm Sign in Scheme 5 on behalf of $SGS_1,SGS_2,\ldots,SGS_{n_S-1}$. It is a valid signature because $g^z=g^{\sum z_i}=\prod g^{z_i}=\prod\left(g^{R_i}\cdot g^{usk_i\cdot h_1}\cdot g^{r_i\cdot(h'_2)^{-1}}\right)=\prod\left(W_i\cdot upk_i^{h_1}\cdot U_i^{h_2}\right)=W\cdot mpk^{h_0}\cdot upk^{h_1}\cdot U^{h_2}$.

According to Theorems 1–3, we obtain the following theorem easily.

Theorem 4. Our CBTS scheme is existentially unforgeable against adaptive chosen message attacks under the DL assumption in the random oracle model.

6. COMPARISON AND APPLICATION

The notion of CBTS is proposed herein for the first time. The proposed notion and scheme of CBTS is based on analyzing the advantages and disadvantages of many existing models and schemes of IDTS and CLTS. We compare them in Table I. From the table, we see that our scheme possesses the following advantages. First, our scheme is more efficient than others owing to the use of DL assumption without pairings. Second, our scheme is more general because both master secret key and user secret key are distributed to corresponding parties. Third, our scheme is more practical because the trusted dealer does not need to distribute shares. Fourth, unlike most existing schemes that require all parties to generate a signature (certificate) in the Sign (Certify) phase, our scheme requires only t_S (t_C) or more than t_S (t_C) parties to generate a

Table I. Comparison of several schemes.

Scheme	Chen *et al.* IDTS [13]	Wang *et al.* CLTS [14]	Yuan *et al.* CLTS [15]	Our CBTS
Mathematic tool	Pairings	Pairings	Pairings	DL assumption
Provable secure	Yes	Yes	Yes	Yes
msk share	No	Yes	No	Yes
usk share	Yes	Yes	Yes	Yes
Dealer	No	Yes	No	No
Signers number	t_S or more	t_S or more	n_S	t_S or more
Detect dishonest	Yes	No	Yes	Yes

signature (certificate). Finally, our scheme can detect dishonest participants because our scheme uses JP-DKG.

Next, we discuss the efficiency of our scheme. In the Wang *et al*. scheme [14], the trusted dealer knows all the shared secret value and can sign a valid signature. Furthermore, the participants cannot detect dishonest participants. These features are different from our proposed scheme. Thus, we compare our scheme with those of Chen *et al*. [13] and Yuan *et al*. [15] only. The computational cost of our scheme is much less than these schemes [13,15] owing to the feature of using the DL assumption without pairings. Furthermore, our scheme requires only a subset of participants to generate a signature; but the scheme in [15] requires all participants to generate a signature, which is impractical in most applications.

We use the following scenaro to illustrate our scheme. Assume that there is a company having n_S directors and some of the directors may act dishonestly in signing a document. The company's policy requires that each document of the company must be signed by t_S or more than t_S directors. The company adopts the CBTS as its signature scheme. In order to prevent the adversary from compromising the master key or performing denial-of-service attacks against the trusted authorities [16], the company adopts n_C CGSs to collectively generate the master secret key, and each CGS has a share of the master secret key. All directors collectively generate the company's secret key and public key and send the public key to CGSs for the certificate. Any t_C or more than t_C CGSs collectively generate the shares of the public key's certificate and send shares to the corresponding director. Later, on behalf of the company, any t_S or more than t_S directors collectively can generate the signature of the message. On the other hand, anyone can verify the validity of the signature. In our proposed scheme, if there is any dishonest director that generates a false value in the Sign phase, the other participating directors can detect this misbehavior. However, the scheme in [14] cannot detect this misbehavior. Thus, the dishonest director can self-generate a valid signature after obtaining valid values from honest directors whereas other honest directors cannot. In addition, the scheme in [15] requires all directors to participate in the Sign phase, which is inefficient.

7. CONCLUSION

In this paper, we propose the notion of CBTS. The model of CBTS is a general model that allows both the master secret key and the user secret key to be split into shares and distributed to corresponding participators. In practical application, we can either split a single key or split both keys into shares and share these among participants. Furthermore, the model can be easily converted into an IDTS model to solve the key escrow problem, or a CLTS model.

Our CBTS scheme is existentially unforgeable against adaptive chosen message attacks under the DL assumption in the random oracle. Compared with existing schemes, our scheme requires neither pairings computation nor a trusted dealer. Furthermore, unlike other schemes that require all entities to jointly generate a signature in the Sign phase, our scheme only requires t (the threshold) or more than t entities to generate a signature. Our scheme can detect dishonest participants through the use of JP-DKG.

REFERENCES

1. Desmedt Y, Frankel Y. Shared generation of authenticators and signatures. In *Proceedings of 11th Annual International Cryptology Conference on Advances in Cryptology - CRYPTO'91*, 1991; 457–469.
2. Shamir A. How to share a secret. *Communications of the ACM* 1979; **22**(11):612–613.
3. Gennaro R, Jarecki S, Krawczyk H, Rabin T. Robust threshold DSS signatures. *Information and Computation* 2001; **164**(1):54–84.
4. Gennaro R, Jarecki S, Krawczyk H, Rabin T. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology* 2007; **20**(1):51–83.
5. Shamir A. Identity based cryptosystems and signature schemes. In *Proceedings of CRYPTO'84 on Advances in Cryptology*, 1984; 47–53.
6. Abelson H, Anderson R, Bellovin SM, *et al*. The Risks of Key Recovery, Key Escrow, and Trusted Third-Party Encryption. http://www.schneier.com/paper-key-escrow.pdf (Accessed April 10, 2013).

7. Al-Riyami SS, Paterson KG. Certificateless public key cryptography. In *Proceedings of 9th International Conference on the Theory and Application of Cryptology and Information Security on Advances in Cryptology - ASIACRYPT'03*, 2003; 452–473.
8. Gentry C. Certificate-based encryption and the certificate revocation problem. In *Proceedings of the 22nd International Conference on Theory and Applications of Cryptographic Techniques on EUROCRYPT'03*, 2003; 272–293.
9. Li JG, Wang ZW, Zhang YC. Provably secure certificate-based signature scheme without pairings. *Information Sciences* 2013. doi:10.1016/j.ins.2013.01.013.
10. Harn L. Group-oriented (*t*, *n*) threshold digital signature scheme and digital multisignature. *IEE Proceedings - Computers and Digital Techniques* 1994; **141**(5):307–313.
11. Kim S, Kim J, Cheon JH, Ju SH. Threshold signature scheme for ElGamal variants. *Computer Standards & Interfaces* 2011; **33**(4):432–437.
12. Baek J, Zheng YL. Identity-based threshold signature scheme from the bilinear pairings. In *Proceedings of the International Conference on Information Technology: Coding and Computing*, 2004; 124–128.
13. Chen XF, Zhang FG, Konidala DM, Kim K. New ID-based threshold signature scheme from bilinear pairings. In *Proceedings in 5th International Conference on Cryptology in India on Progress in Cryptology - INDOCRYPT'04*, 2004; 371–383.
14. Wang LC, Cao ZF, Li XX, Qian HF. Simulatability and security of certificateless threshold signatures. *Information Sciences* 2007; **177**(7):1382–1394.
15. Yuan H, Zhang FT, Huang XY, Mu Y, Susilo W, Zhang L. Certificateless threshold signature scheme from bilinear maps. *Information Sciences* 2010; **180**(23):4714–4728.
16. Lu Y, Li JG, Xiao JM. Threshold certificate-based encryption: definition and concrete construction. In *Proceedings of the 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, 2009; 278–282.
17. Feldman P. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science - FOCS'87*, 1987; 427–438.
18. Pedersen TP. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology - CRYPTO'91*, 1991; 129–140.