Modification and Optimisation of an ElGamal-Based PVSS Scheme

Kun Peng

Institute for Infocomm Research, Singapore
dr.kun.peng@gmail.com

Abstract. Among the existing PVSS schemes, a proposal by Shoemakers is a very special one. It avoids a common problem in PVSS design and costly operations by generating the secret to share in a certain way. Although its special secret generation brings some limitations to its application, its improvement in simplicity and efficiency is significant. However, its computational cost is still linear in the square of the number of share holders. Moreover, appropriate measures need to be taken to extend its application. In this paper, the PVSS scheme is modified to improve its efficiency and applicability. Firstly, a more efficient proof technique is designed to reduce the computational cost of the PVSS scheme to be linear in the number of share holders. Secondly, its secret generation procedure is extended to achieve better flexibility and applicability.

1 Introduction

In many secure information systems, some secret information is distributed among multiple parties to enhance security and robustness of systems. The first secret sharing technique is Shamir's t-out-of-n secret sharing [22]. A dealer has a secret s and wants to share it among n share holders. The dealer builds a polynomial $f(x) = \sum_{j=0}^{t-1} \alpha_j x^j$ and sends $s_i = f(i) \bmod q$ to the i^{th} share holder for $i = 1, 2, \ldots, n$ through a secure communication channel where $\alpha_0 = s$ and q is an integer larger than any possible secret to share. Any t shares can be used to reconstruct the secret $s = \sum_{i \in S} s_i u_i \bmod q$ where $u_i = \prod_{j \in S, j \neq i} j/(j-i) \bmod q$ and S contains the indices of the t shares. Moreover, no information about the secret is obtained if the number of available shares is less than t. Secret sharing is widely employed in various secure information systems like e-auction, e-voting and multiparty computation. As most of the applications require public verifiability, very often secret sharing must be publicly verifiable. Namely, it must be publicly verified that all the n shares are consistently generated from a unique share generating polynomial and any t of them reconstruct the same secret. As the verification is public in those distributed systems, any wrong-doing can be publicly detected by any one and thus is undeniable.

Publicly verifiable secret sharing is usually called PVSS. It is widely employed in various applications like mix network [1], threshold access control [20], e-voting [9,12,14,13], distributed encryption algorithm [6], zero knowledge proof [7], anonymous token [11] and fair exchange [15]. In PVSS the dealer is not

A. Youssef, A. Nitaj, A.E. Hassanien (Eds.): AFRICACRYPT 2013, LNCS 7918, pp. 311–327, 2013.

trusted and may deviate from the secret sharing protocol. He may distribute inconsistent shares to the share holders such that some shares reconstruct a secret while some other shares reconstruct a different secret. This cheating behaviour compromises security of the applications of secret sharing. For example, in secret-sharing-based multiple-tallier e-voting [9,12], it allows the talliers to tamper with the votes. Famous PVSS schemes include [23,4,10,21], where one of the two PVSS protocols in [23] is a development of the proposal in [8] and [17] is a revisit to [4] with a different encryption algorithm and stronger security. Usually, PVSS is a combination of secret sharing and publicly verifiable encryption of an encryption algorithm, which is employed to implement the secure communication channel for share distribution. The dealer encrypts shares for the share holders using their public keys and publishes the ciphertexts. Each share holder can decrypt the ciphertext for him and obtain his share, while the dealer can publicly prove that the messages encrypted in the ciphertexts are shares for a unique secret without revealing the secret or its shares. The following three important security properties are desired in PVSS.

- Correctness: if the dealer is honest and does not deviate from the PVSS protocol, he can always successfully prove validity of the shares.
- Soundness: only with an exception of a negligible probability, the shares are guaranteed to be generated by the same secret-generating polynomial such that any t of them reconstruct the same secret.
- Privacy: no information about the secret or any of its shares is revealed in the proof of validity of the shares. More precisely, a private PVSS scheme should employ zero knowledge proof techniques, which do not reveal any secret information as their proof transcripts can be simulated without any difference by a party without any knowledge of the secret or any of its shares.

Those security properties can be defined in a fornal way as follows.

Definition 1 *There is a proof function $Val(s_1, s_2, \ldots, s_n)$ for a dealer to prove validity of the shares. If his proof returns $Val(s_1, s_2, \ldots, s_n) = TRUE$, validity of $s_1, s_2, \ldots, s_n$ are accepted. If it returns $FALSE$, the shares are regarded as invalid.*

- *Correctness: an honest dealer can always achieve $Val(s_1, s_2, \ldots, s_n) = TRUE$ and $Pr(Val(s_1, s_2, \ldots, s_n) = FALSE$ and $\exists f(x) = \sum_{j=0}^{t-1} \alpha_j x^j$ such that $s_i = f(i) \bmod q) = 0$ where $Pr()$ stands for the probability of an event.*
- *Soundness: $Pr(Val(s_1, s_2, \ldots, s_n) = TRUE$ and $\nexists f(x) = \sum_{j=0}^{t-1} \alpha_j x^j$ such that $s_i = f(i) \bmod q)$ is negligible.*
- *Privacy: suppose the transcript of $Val(s_1, s_2, \ldots, s_n)$ is $TRANS$, and then anyone without knowledge of s or any share can generate a simulating transcript $TRAN'$ such that distribution of the two transcripts is indistinguishable.*

Although public verification of some operations is discussed in a VSS (verifiable secret sharing) scheme [18], it is not a PVSS proposal as it does not implement publicly verifiable encryption.

There are two basic and compulsory requirements in PVSS, security of encryption algorithm and publicly verifiable encryption, which are explained as follows.

- A secure encryption algorithm must be employed for the dealer to encrypt the shares before distributing them to the share holders. The employed encryption algorithm protect privacy of the encrypted messages such that no polynomial adversary can obtain any information about the secret or any of its shares from the ciphertexts of all the shares. This assumption is called *basic* assumption as it is inevitable in any PVSS scheme.
- The employed encryption algorithm must support publicly verifiable encryption such that an encrypted share can be publicly proved and verified to be generated by a secret share-generating polynomial. The existing PVSS schemes specify the public proof and verification as follows.
  1. The dealer publishes $c_i = E_i(s_i)$ for $i = 1, 2, \ldots, n$ where $E_i()$ is the encryption function of the i^{th} share holder's encryption algorithm.
  2. An integer g with multiplicative order q is chosen.
  3. $A_j = g^{\alpha_j}$ for $j = 0, 1, \ldots, t-1$ are published by the dealer as a public commitment to the share-generating polynomial. To enhance privacy, α_j can be committed to in $g^{\alpha_j} h^{r_j}$ where h is in the cyclic group generated by g, $\log_g h$ is secret and r_j is a random integer smaller than the order of g. For simplicity of description, we only discuss the simpler commitment algorithm, while replacing it with the more complex commitment algorithm does not change the specification of PVSS in essence.
  4. A commitment to every share s_i is publicly available: $C_i = \prod_{j=0}^{t-1} A_j^{i^j}$.
  5. The dealer has to publicly prove that the same integer is committed to in C_i and encrypted in c_i.

1.1 A Dilemma in Choosing Encryption Algorithm

Choice of encryption algorithm is a subtle question in PVSS. Most existing PVSS schemes [4,10,17] choose RSA or Paillier encryption [16] for the share holders. Those two encryption algorithms have a common property: suppose the decryption function of the i^{th} share holder's encryption algorithm is $D_i()$ and the modulus used in the calculations in $D_i()$ is q_i and then $q_1, q_2, \ldots, q_n$ must be different from each other and so cannot be equal to q. More precisely, $D_i(E_i(m))$ is the remainder of m modulo q_i and is not necessary to be m mod q as at least $n-1$ of the q_is cannot be equal to q. So if a dealer encrypts a message larger than q_i in c_i what the i^{th} share holder obtains through $D_i(c_i)$ is not equal to what the dealer encrypts in c_i modulo q. Therefore, a malicious dealer can cheat as follows.

1. He publishes $c_i = E_i(f(i)+kq)$ where k is an integer to satisfy $f(i)+kq > q_i$.
2. The i^{th} share holder obtains $D_i(c_i) = f(i) + kq \bmod q_i$, which is not equal to $f(i)$ modulo q.
3. The dealer can still prove that the same integer (namely $f(i) + kq$) is committed to in C_i and encrypted in c_i. But $D_i(c_i)$ cannot be used as a share to reconstruct s, the secret committed to in C_0.

The simplest way to prevent this attack is to set $q_1 = q_2 = \ldots = q_n = q$. However this setting is impossible with RSA or Paillier encryption. So in the PVSS schemes employing RSA or Paillier encryption [4,10,17], the dealer has to prove that the message committed to in C_i and encrypted in c_i is smaller than q_i for $i = 1, 2, \ldots, n$. Therefore, n instances of range proof is needed where a range proof proves that a secret integer is within a certain range without revealing any other information about it. The most efficient general-purpose range proof is proposed by Boudot in Section 3 of [5]. Although the range proof by Boudot costs only a constant number of exponentiations, it is still not efficient enough. His range proof consists of two proof operations to handle the upper bound and lower bound of the range respectively, while each of them costs about 20 exponentiations. n instances of such range proof in PVSS is a high cost. Range proof can be more efficient in terms of the number of exponentiations when a secret integer is chosen from a range and then proved to be in a much larger range. This condition is called expansion rate and explained in details in Section 1.2.3 of [5]. For range proof in a range R, a much smaller range R' is chosen in the middle of R. The prover chooses a message v in R' and publishes its monotone function $z = w + cv$ in Z where w is randomly chosen from a set S_1 and c is randomly chosen from a set S_2. Then z is verified to be in a range R''. Of course, both v and w must be sealed in some appropriate commitments (or ciphertexts) and satisfaction of $z = w + cv$ is proved by appropriately processing the commitments without revealing v or w. More details of this efficient special range proof (e.g. how to set the sizes of the ranges and sets) can be found in Section 1.2.3 of [5], which calls it CFT proof and shows that when the parameters are properly chosen v can be guaranteed to be in R with an overwhelmingly large probability. Obviously, this range proof is a special solution instead of a general-purpose range proof technique. Usage of this method is not easy and liable to many limitations. So its application should be cautious. For example, R must be at least billions of times larger than R'. Moreover, to minimize the information about v revealed from its monotone function[1] z, w must be at least billions of times larger than cv. In addition, for soundness of range proof, the relation between R'' and the other parameters must be delicately designed. As a result, R'' is usually very large and contains extremely large integers. Extra large integers and computation of them in Z without any modulus bring additional cost and should be taken into account in efficiency analysis.

It is easy to notice that ElGamal encryption supports the simple solution $q_1 = q_2 = \ldots = q_n = q$. However, it is not easy to prove that the message in an ElGamal ciphertext is committed to in a commitment as the exponent to a public base. The first PVSS scheme employing ElGamal encryption [23] uses the cut-and-choose strategy and needs to repeat its proof quite a few times to guarantee that the same integer is committed to in C_i and encrypted in c_i with a large enough probability. So it is inefficient. This dilemma in choice of encryption algorithm is partially solved in [21]. On one hand it employs ElGamal encryption

[1] Revealing of secret information is inevitable as z is calculated in Z as a monotone function of v.

and sets $q_1 = q_2 = \ldots = q_n = q$ and so avoids range proof of the shares. On the other hand, it generates the secret to share in a special way to avoid the cut-and-choose proof mechanism. However, in the PVSS scheme in [21], the dealer must know the discrete logarithm of the secret to share to a public base and generate the secret from the discrete logarithm. As explained in section 2.1, this special secret generation procedure limits application of the PVSS scheme (e.g. in circumstances where the discrete logarithm of the secret to share is unknown).

1.2 Our Contribution

The analysis above has shown that the RSA and Paillier based PVSS schemes [4,10,17] depend on n instances of range proof and ElGamal-based PVSS schemes [23,21] have their own drawbacks like complex and costly proof mechanism and limited application area. Another point we need to mention is that even the PVSS scheme in [21] is not efficient enough and cost $O(tn)$ in computation. Our task is to overcome all those drawbacks.

In this paper, the PVSS scheme in [21] is modified and optimised to achieve higher efficiency and better applicability. The new PVSS scheme sets $q_1 = q_2 = \ldots = q_n = q$ and thus is inherently free of any range proof. It not only avoids cut-and-choose in proof of validity of secret sharing but also achieves much higher efficiency than any existing PVSS scheme. It only costs $O(n)$ in computation. Moreover, it addresses the problem of limited application caused by the special secret generation mechanism in [21] and provides an alternative solution. Another advantage of our new PVSS scheme over the existing PVSS schemes is that it only needs the basic assumption and a verifier in zero knowledge proof (who generates random challenges and can be replaced by a (pseudo)random function in the random oracle model), both of which are inevitable in any PVSS scheme. In comparison, the existing PVSS schemes need to publish a public commitment to the share-generating polynomial as recalled earlier in this section. As no commitment algorithm is completely assumption-free and any commitment algorithm needs some computational assumption to guarantee that the secret message in the commitment is private and cannot be changed, they may need some additional computational assumption(s)[2].

2 Background and Preliminaries

Background knowledge and security model are given in this section.

2.1 The PVSS Scheme by Schoenmakers

In the PVSS scheme in [21], a dealer employs a special sharing function to share a specially-generated secret, while the other PVSS schemes employs the normal

[2] Actually, most of the existing PVSS schemes do need some additional computational assumptions.

sharing function by Shamir [22] and can share any secret in general. Moreover, a corresponding special secret reconstruction function is employed in [21] to reconstruct the secret accordingly. The special sharing function reconstruction function in [21] are described as follows while its public proof of validity of secret sharing follows the principle recalled in Section 1, which is commitment-based and applies to all the existing PVSS.

- Setting
  p and q are large primes and q is a factor of $p-1$. G_q is the cyclic group with order q in Z_p^* and g is a generator of G_q.
- Sharing
  1. The dealer firstly chooses δ and then calculates the secret $s = g^{\delta}$.
  2. He builds a polynomial $f(x) = \sum_{j=0}^{t-1} a_j x^j$ with $a_0 = \delta$ and a_j for $j = 1, 2, \ldots, t-1$ are random integers.
  3. He publishes ElGamal ciphertext $c_i = (g^{r_i},\ g^{\delta_i} y_i^{r_i})$ for $i = 1, 2, \ldots, n$ where $\delta_i = f(i)$, y_i is P_i's ElGamal public key.
- Reconstruction
  1. Each P_i decrypt c_i and obtains $s_i' = g^{\delta_i}$.
  2. A set with at least t sharers are put together: $s = \prod_{i \in I} {s'}_i^{u_i}$ where $u_i = \prod_{j \in I, j \neq i} j/(j-i)$ and I is the set containing the indices of the t shares.

In the PVSS scheme in [21], $a_0 = \delta$, so actually discrete logarithm of the secret s is shared using the share-generating polynomial. Its reconstruction function is accordingly changed to reconstruct s using the shares of δ. So knowledge of discrete logarithm of the secret is compulsory to the dealer and due to hardness of DL problem its discrete logarithm must be fixed before the secret is generated in the PVSS scheme in [21]. Therefore, it is not suitable for some applications. One of the most common applications of PVSS is key sharing (or called distributed key generation in some cryptographic schemes). A secret key is usually chosen from a consecutive interval range instead of a cyclic group in many encryption algorithms (e.g. AES and normal ElGamal) and many other encryption algorithms (e.g. RSA and Paillier) do not first choose a discrete logarithm of the secret key and then calculate the secret key in a cyclic group either. Another important application of PVSS is sharing of password or accessing code in distributed access control. In most applications, a password or accessing code is usually randomly chosen by users and very often the users would like to choose some special password or accessing code like their birthdays or phone number. So it is very probable that discrete logarithm of the password or accessing code is unknown or even does not exist at all. In general, the secret to share cannot be generated in the discrete-logarithm-fixed-first manner in many applications. To guarantee that the secret generated in the discrete-logarithm-fixed-first manner is distributed in a distribution in a set R required by an application, the set $R' = \{x \mid g^x \in R\}$ must be calculated and thus discrete logarithm of the secret can be chosen from it first. However, due to hardness of DL problem, it is often hard to calculate R'.

In summary, although the PVSS scheme in [21] employs ElGamal encryption to avoid range proof and other complex proof operations for the first time and partially solved the dilemma explained in Section 1.1, it still has some drawbacks. Firstly, it still needs $O(tn)$ in computation and is thus not efficient enough. Secondly, its has some limitations in practical application.

2.2 Security Model

Soundness of PVSS is defined in Definition 2, while privacy of PVSS follows the simulation-based general definition of privacy of any proof protocol in Definition 3.

Definition 2 *(Soundness of PVSS). If the dealer's public proof of validity of the shares passes the public verification of a sound PVSS with a non-negligible probability, there exist integers* $\alpha_0, \alpha_1, \ldots, \alpha_{t-1}$ *such that* $s_i = \sum_{j=0}^{t-1} \alpha_j i^j$ *for* $i = 1, 2, \ldots, n$ *where* s_i *is the* i^{th} *share.*

Definition 3 *(Privacy of a proof protocol). A proof protocol is private if its transcript can be simulated by a party without any knowledge of any secret such that the simulating transcript is indistinguishable from the real protocol transcript as defined in Definition 4.*

Definition 4 *(Indistinguishability of distributions). Suppose a set of variables have two transcripts respectively with two distributions* T_1 *and* T_2*. A random secret bit* i *is generated. If* $i = 0$*, two instances of* T_2 *are published; If* $i = 1$*, an instance of* T_1 *and an instance of* T_2 *are published. An algorithm can distinguish* T_1 *and* T_2 *if given the two published instances of transcripts he can calculate* i *with a probability non-negligibly larger than 0.5.*

3 New PVSS Based on ElGamal Encryption

Our new PVSS protocol employs ElGamal encryption algorithm to maintain consistency of modulus and avoid range proof like in [21] but uses a completely different proof technique. It does not employ any commitment algorithm to avoid additional computational assumption. Of course it still needs the basic assumption, namely semantic security of ElGamal encryption defined in the following.

Definition 5 *(Semantic security of encryption algorithm) A polynomial adversary chooses two messages* m_0 *and* m_1 *in the message space of the encryption algorithm in any way he likes and submits them to an encryption oracle, who randomly chooses a bit* i *and returns the adversary* $E(m_i)$ *where* $E()$ *denotes the encryption algorithm. The encryption algorithm is semantically secure if the probability that the adversary obtains* i *is not non-negligibly larger than 0.5.*

The main idea of the new PVSS technique is simple: given two sets of shares $s_1, s_2, \ldots, s_n$ and $k_1, k_2, \ldots, k_n$ and a random challenge R, if $k_1 + Rs_1, k_2 + Rs_2, \ldots, k_n + Rs_n$ are a set of consistent shares of the same secret, $s_1, s_2, \ldots, s_n$ are a set of consistent shares of the same secret with an overwhelmingly large probability. It is described as follows where the denotations in [21] are inherited and the share holders are $P_1, P_2, \ldots, P_n$.

1. The dealer firstly chooses δ in Z_q and then calculates $s = g^\delta \bmod p$. In this way, he generates an s in G_q such that he knows $\delta = \log_g s$.
2. He builds a polynomial $f_1(x) = \sum_{j=0}^{t-1} \alpha_j x^j$ with $\alpha_0 = \delta$ and α_j for $j = 1, 2, \ldots, t-1$ are random integers chosen from Z_q.
3. He randomly chooses ϵ in Z_q and builds a polynomial $f_2(x) = \sum_{j=0}^{t-1} \beta_j x^j$ with $\beta_0 = \epsilon$ where β_j for $j = 1, 2, \ldots, t-1$ are random integers chosen from Z_q.
4. He publishes $c_i = (g^{r_i} \bmod p,\ g^{\delta_i} y_i^{r_i} \bmod p)$ and $c'_i = (g^{r'_i} \bmod p,\ g^{\epsilon_i} y_i^{r'_i} \bmod p)$ for $i = 1, 2, \ldots, n$ where $\delta_i = f_1(i) \bmod q$, $\epsilon_i = f_2(i) \bmod q$ and r_i, r'_i are randomly chosen from Z_q.
5. A random challenge R in Z_q is publicly generated by one or more verifier(s) or a (pseudo)random function like in any publicly verifiable zero knowledge proof protocols (e.g. those zero knowledge proof primitives used in all the existing PVSS schemes). A simple and non-interactive method to publicly generate R is $R = H(c_1, c'_1, c_2, c'_2, \ldots, c_n, c'_n)$ where $H()$ is a (pseudo)random function (e.g. a hash function). As mentioned in Section 1.2, this randomness generation procedure is the same as the randomness generation procedure necessary in any existing PVSS scheme and needs no additional assumption or technique.
6. He publishes $\gamma_j = \beta_j + R\alpha_j \bmod q$ for $j = 0, 1, \ldots, t-1$ and $R_i = Rr_i + r'_i \bmod q$ for $i = 1, 2, \ldots, n$ and any one can publicly verify

$$a_i^R a'_i = g^{R_i} \bmod p \tag{1}$$

$$b_i^R b'_i = g^{S_i} y_i^{R_i} \bmod p \tag{2}$$

   for $i = 1, 2, \ldots, n$ where $c_i = (a_i, b_i)$, $c'_i = (a'_i, b'_i)$ and

$$S_i = \sum_{j=0}^{t-1} \gamma_j i^j \bmod q. \tag{3}$$

7. Share decryption and secret reconstruction are as follows.
   (a) Each P_i decrypts c_i and obtains $s_i = b_i / a_i^{x_i} \bmod p$.
   (b) A set with at least t sharers are put together: $s = \prod_{i \in S} s_i^{u_i} \bmod p$ where $u_i = \prod_{j \in S, j \neq i} j/(j-i) \bmod q$ and S is the set containing the indices of the t shares.

Soundness and privacy of the new ElGamal-based PVSS protocol are proved in Theorem 1 and Theorem 2 respectively, following Definition 2 and Definition 3 respectively.

Theorem 1. *Proof: If Equations (1), (2) and (3) hold for $i = 1, 2, \ldots, n$ with a probability larger than $1/q$, then there exists a t-out-of-n share-generating polynomial $f()$ such that $\log_g D_i(c_i) = f(i) \bmod q$ for $i = 1, 2, \ldots, n$ where $D_i()$ denotes the ElGamal decryption function of P_i.*

Equations (1), (2) and (3) for $i = 1, 2, \ldots, n$ with a probability larger than $1/q$ imply

$$D_i(c_i^R c_i') = g^{S_i} = g^{\sum_{j=0}^{t-1} \gamma_j i^j} \bmod p \text{ for } i = 1, 2, \ldots, n \tag{4}$$

with a probability larger than $1/q$. So, there must exist two different integers in Z_q, R and R', such that the dealer can produce $\gamma_0, \gamma_1, \ldots, \gamma_{t-1}$ and $\gamma_0', \gamma_1', \ldots, \gamma_{t-1}'$ respectively to satisfy

$$D_i(c_i^R c_i') = g^{\sum_{j=0}^{t-1} \gamma_j i^j} \bmod p \text{ for } i = 1, 2, \ldots, n \tag{5}$$

$$D_i(c_i^{R'} c_i') = g^{\sum_{j=0}^{t-1} \gamma_j' i^j} \bmod p \text{ for } i = 1, 2, \ldots, n. \tag{6}$$

Otherwise, there is at most one R in Z_q for the dealer to produce a set of integers $\gamma_0, \gamma_1, \ldots, \gamma_{t-1}$ to satisfy (4) and the probability that (4) is satisfied is no larger than $1/q$, which is a contradiction.

(5) divided by (6) yields

$$D_i(c_i)^{R-R'} = g^{\sum_{j=0}^{t-1} (\gamma_j - \gamma_j') i^j} \bmod p \text{ for } i = 1, 2, \ldots, n.$$

Namely,

$$\log_g D_i(c_i)^{(R-R')} = \textstyle\sum_{j=0}^{t-1} (\gamma_j - \gamma_j') i^j \bmod q \text{ for } i = 1, 2, \ldots, n$$

and thus

$$\log_g D_i(c_i) = \sum_{j=0}^{t-1} ((\gamma_j - \gamma_j')/(R - R')) i^j \bmod q \text{ for } i = 1, 2, \ldots, n. \qquad \square$$

Theorem 2. *Privacy of the new ElGamal-based PVSS protocol is completely achieved, only dependent on the basic assumption and randomness of the challenge R. More precisely, their privacy is formally provable on the assumption that the employed ElGamal encryption algorithm is semantically secure and R is random.*

Proof: Both protocols have the same PVSS transcript $R, c_1, c_2, \ldots, c_n, c_1', c_2', \ldots, c_n', R_1, R_2, \ldots, R_n, \gamma_0, \gamma_1, \ldots, \gamma_{t-1}$. So their privacy can be universally proved in one simulation. A party without any access to the secret or any of its shares can simulate the PVSS transcript and generate a simulated transcript as follows.

1. R, $\alpha_0, \alpha_1, \ldots, \alpha_{t-1}$, $\beta_0, \beta_1, \ldots, \beta_{t-1}$, $r_1, r_2, \ldots, r_n$, $r_1', r_2', \ldots, r_n'$ are randomly chosen from Z_q.

2. $R_i = r'_i + Rr_i \bmod q$ for $i = 1, 2, \ldots, n$.
3. $\gamma_j = R\alpha_j + \beta_j \bmod q$ for $j = 0, 1, \ldots, t-1$.
4. $s_i = \sum_{j=0}^{t-1} \alpha_j i^j \bmod q$ for $i = 1, 2, \ldots, n$.
5. $k_i = \sum_{j=0}^{t-1} \beta_j i^j \bmod q$ for $i = 1, 2, \ldots, n$.
6. $c_i = (g^{r_i} \bmod p,\ g^{s_i} y_i^{r_i} \bmod p)$ for $i = 1, 2, \ldots, n$.
7. $c'_i = (g^{r'_i} \bmod p,\ g^{k_i} y_i^{r'_i} \bmod p)$ for $i = 1, 2, \ldots, n$.

In this simulated transcript of c_i, c'_i, R_i for $i = 1, 2, \ldots, n$, $R, \gamma_0, \gamma_1, \ldots, \gamma_{t-1}$,

- each of c_i and c'_i alone is uniformly distributed in the ciphertext space of the employed ElGamal encryption algorithm for $i = 1, 2, \ldots, n$;
- each of $R, \gamma_0, \gamma_1, \ldots, \gamma_{t-1}$ is uniformly distributed in Z_q;
- each R_i is uniformly distributed in Z_q for $i = 1, 2, \ldots, n$;
- $D_1(c_1), D_2(c_2), \ldots, D_n(c_n)$ are shares of an integer uniformly distributed in G_q;
- $D_1(c'_1), D_2(c'_2), \ldots, D_n(c'_n)$ are shares of an integer uniformly distributed in G_q.

In comparison, in the real transcript of c_i, c'_i, R_i for $i = 1, 2, \ldots, n$, $R, \gamma_0, \gamma_1, \ldots, \gamma_{t-1}$ published by the dealer,

- each of c_i and c'_i alone is uniformly distributed in the ciphertext space of the employed ElGamal encryption algorithm for $i = 1, 2, \ldots, n$;
- each of $R, \gamma_0, \gamma_1, \ldots, \gamma_{t-1}$ is uniformly distributed in Z_q;
- each R_i is uniformly distributed in Z_q for $i = 1, 2, \ldots, n$;
- $D_1(c_1), D_2(c_2), \ldots, D_n(c_n)$ are shares of the shared secret s or $g^s \bmod p$, depending on which of the two protocols is referred to;
- $D_1(c'_1), D_2(c'_2), \ldots, D_n(c'_n)$ are shares of an integer uniformly distributed in G_q.

The only difference between the two transcripts lies in distribution of $c_1, c_2, \ldots, c_n$, which are encrypted shares of a random integer in the simulated transcript and encrypted shares of s or $g^s \bmod p$ in the real transcript. The transcript of $c_1, c_2, \ldots, c_n$ in the real ElGamal-based new PVSS protocols is denoted as T_1; while the simulated transcript of $c_1, c_2, \ldots, c_n$ is denoted as T_2. If an algorithm can compromise privacy of the ElGamal-based new PVSS protocols, according to Definition 3, it can distinguish T_1 and T_2 as defined in Definition 4. This algorithm, denoted as A, can be employed to win a game as follows.

1. An adversary sets $m_0 = s$ and randomly chooses m_1 from G_q and submits them to a dealer. The dealer randomly chooses a bit i and shares m_i among the share holders using an ElGamal-based new PVSS protocol. The party needs to calculate i to win the game using the encrypted shares, which is denoted as T'_1. Note that T'_1 and T_1 have the same distribution if $i = 0$; while T'_1 and T_2 have the same distribution if $i = 1$.
2. The adversary randomly chooses an integer from G_q and shares it among the share holders using the ElGamal-based new PVSS protocol. He generates the encrypted shares of this integer, which is denoted as T'_2. T'_2 and T_2 have the same distribution.

3. He inputs T_1' and T_2' to A, which outputs i'. As A can distinguish T_1 and T_2 as defined in Definition 4, with a probability non-negligibly larger than 0.5, it returns $i' = 0$ if $i = 1$ and returns $i' = 1$ if $i = 0$. So $1 - i'$ is a correct solution for i with a probability non-negligibly larger than 0.5.

However, to win the game with a probability non-negligibly larger than 0.5 will leads to a contradiction as follows. For simplicity of proof, suppose $2t \geq n + 1$ and like many existing PVSS descriptions the addition modulus is not explicitly specified. If there is an polynomial algorithm A to win the game above, it is illustrated in the following that this algorithm can be employed to break semantic security of the employed encryption algorithm.

1. The adversary in Definition 5 needs to obtain i where the encryption algorithm is $E()$.
2. He calculates integers $\alpha_0, \alpha_1, \ldots, \alpha_{t-1}$, $\alpha_0', \alpha_1', \ldots, \alpha_{t-1}'$ and $s_2, s_3, \ldots, s_n$ such that

$$m_0 = \textstyle\sum_{j=0}^{t-1} \alpha_j$$
$$m_1 = \textstyle\sum_{j=0}^{t-1} \alpha_j'$$
$$s_i = \textstyle\sum_{j=0}^{t-1} \alpha_j i^j \text{ for } i = 2, 3, \ldots, n$$
$$s_i = \textstyle\sum_{j=0}^{t-1} \alpha_j' i^j \text{ for } i = 2, 3, \ldots, n.$$

   As $2t \geq n + 1$, he can always find such integers using efficient linear algebra computations.
3. He inputs (α_0, α_0') to A as the two possible secrets to share. He inputs $(c, c_2, c_3, \ldots, c_n)$ to A as the encrypted shares where $c_i = E_i(s_i)$ for $i = 2, 3, \ldots, n$ and each $E_i()$ denotes the same type of encryption algorithm as $E()$ but with a different key.
4. A guesses which secret is shared in the encrypted shares. As A can break semantic security of CCSD of the PVSS protocol, the probability that it gets a correct guess is P, which is non-negligibly larger than 0.5.
5. If A returns 0, the adversary outputs $i = 0$; if A returns 1, he outputs $i = 1$. Note that
   - if $c = E(m_0)$, then $(c, c_2, c_3, \ldots, c_n)$ are encrypted shares of α_0;
   - if $c = E(m_1)$, then $(c, c_2, c_3, \ldots, c_n)$ are encrypted shares of α_0'.

   So the probability that the adversary obtains i is P and non-negligibly larger than 0.5.

Therefore, semantic security of $E()$ is broken. When $2t < n + 1$ the proof can work as well but in a more complex way as sometimes neither of the two possible secrets is shared in the encrypted shares and A has to return a random output. Due to space limit, this extension is left to interested readers.

Since a contradiction is found, T_1 and T_2 cannot be can distinguished in polynomial time. Therefore, if ElGamal encryption is semantical secure, the ElGamal-based new PVSS protocol achieve privacy. □

4 Broader Range of Applications and Batch Verification

Application of our new PVSS technique to a broader range and its efficiency improvement through batch verification are discussed in this section.

4.1 When the Logarithm-Known Secret Generation Mechanism Cannot Work

If s is not a random string to be freely appointed, but a secret with real meaning (e.g. a vote in an e-voting application or a bid in an e-auction system), the PVSS protocol with the logarithm-known secret generation mechanism in last section cannot work. However, a secret with real meaning is usually distributed in a relatively small space. For example, a vote is usually the name of one of the candidates and a bid is usually a biddable price. In this case, searching for discrete logarithm in the small space is not too costly (e.g. using Pollard's Lambda method) and thus our PVSS design can be slightly modified as follows to solve the problem.

1. To share a secret s, the dealer builds a polynomial $f_1(x) = \sum_{j=0}^{t-1} \alpha_j x^j$ with $\alpha_0 = s$ and α_j for $j = 1, 2, \ldots, t-1$ are random integers chosen from Z_q.
2. He randomly builds another polynomial $f_2(x) = \sum_{j=0}^{t-1} \beta_j x^j$ where β_j for $j = 0, 1, \ldots, t-1$ are random integers chosen from Z_q.
3. He publishes $c_i = (g^{r_i} \bmod p,\ g^{s_i} y_i^{r_i} \bmod p)$ and $c_i' = (g^{r_i'} \bmod p,\ g^{k_i} y_i^{r_i'} \bmod p)$ where $s_i = f_1(i) \bmod q$, $k_i = f_2(i) \bmod q$ and r_i, r_i' are randomly chosen from Z_q.
4. A random challenge R in Z_q is generated in the same way as in Section 3.
5. He publishes $\gamma_j = \beta_j + R\alpha_j \bmod q$ for $j = 0, 1, \ldots, t-1$ and $R_i = Rr_i + r_i' \bmod q$ for $i = 1, 2, \ldots, n$ and any one can check Equations (1), (2) and (3) for $i = 1, 2, \ldots, n$.
6. Share decryption and secret reconstruction are as follows.
   (a) Each P_i decrypt c_i and obtains $s_i' = b_i / a_i^{x_i} \bmod p$.
   (b) A set with at least t shares are put together: $s' = \prod_{i \in S} s_i'^{u_i} \bmod p$ where $u_i = \prod_{j \in S, j \neq i} j/(j-i) \bmod q$ and S is the set containing the indices of the t shares.
   (c) $\log_g s'$ is searched for in the space of the secret and the found discrete logarithm is the reconstruction result. As stated before, a space containing practical secret with real meaning is often not too large, so the search is often affordable.

If the secret to share is chosen from a large space and its discrete logarithm is unknown, its discrete logarithm can be calculated in advance to save real-time cost and then the first ElGamal-based PVSS protocol is employed to share the secret.

4.2 PVSS with Explicit Commitment

Although we have shown that in PVSS explicit commitments can be avoided to improve efficiency, some applications of PVSS may need an explicit commitment to the secret, especially when the shared secret has to be processed in the applications without being revealed. In such applications, our new PVSS scheme can be extended to support explicit commitment to the shared secret without compromising high efficiency. It is not necessary to commit to all the coefficients of the share-generating polynomial like in the existing PVSS schemes. Instead, we only commit to the shared secret using a simple commitment mechanism as follows.

1. The secret s is committed to in a public commitment $C = g^s h^\sigma \bmod p$ where σ is randomly chosen from Z_q and h is an integer in G_q such that $\log_g h$ is unknown.
2. When β_0 is employed in PVSS, it is committed to in the same way, namely in a public commitment $C' = g^{\beta_0} h^{\sigma'} \bmod p$ where σ' is randomly chosen in Z_q.
3. When R and γ_0 are published in PVSS, the dealer publishes $\tau = R\sigma + \sigma' \bmod q$ as well. Anyone can publicly verify $C^R C' = g^{\gamma_0} h^\tau \bmod p$ to be ensured that the secret committed in C is shared among the share holders with an overwhelmingly large probability as illustrated in Theorem 3.

Theorem 3. *If Equations (1), (2) and (3) hold for $i = 1, 2, \ldots, n$ and the dealer can calculate $\log_h(C^R C'/g^{\gamma_0})$ with a probability larger than $1/q$, he must have committed to the shared secret in C.*

Proof: Equations (1), (2) and (3) for $i = 1, 2, \ldots, n$ with a probability larger than $1/q$ imply

$$D_i(c_i^R c_i') = g^{S_i} = g^{\sum_{j=0}^{t-1} \gamma_j i^j} \bmod p \text{ for } i = 1, 2, \ldots, n$$

with a probability larger than $1/q$. So, there must exist two different integers in Z_q, R and R', such that the dealer can produce $\gamma_0, \gamma_1, \ldots, \gamma_{t-1}, \tau$ and $\gamma_0', \gamma_1', \ldots, \gamma_{t-1}', \tau'$ respectively to satisfy

$$D_i(c_i^R c_i') = g^{\sum_{j=0}^{t-1} \gamma_j i^j} \bmod p \text{ for } i = 1, 2, \ldots, n \quad (7)$$

$$D_i(c_i^{R'} c_i') = g^{\sum_{j=0}^{t-1} \gamma_j' i^j} \bmod p \text{ for } i = 1, 2, \ldots, n \quad (8)$$

$$C^R C' = g^{\gamma_0} h^\tau \bmod p \quad (9)$$

$$C^{R'} C' = g^{\gamma_0'} h^{\tau'} \bmod p. \quad (10)$$

Otherwise, there is at most one R in Z_q for the dealer to produce $\gamma_0, \gamma_1, \ldots, \gamma_{t-1}, \tau$ to satisfy $D_i(c_i^R c_i') = g^{\sum_{j=0}^{t-1} \gamma_j i^j} \bmod p$ for $i = 1, 2, \ldots, n$ and $C^R C' = g^{\gamma_0} h^\tau \bmod p$ and the probability that he can produce correct responses to satisfy the two equations is no larger than $1/q$, which is a contradiction.

(7)/(8) yields

$$D_i(c_i)^{R-R'} = g^{\sum_{j=0}^{t-1}(\gamma_j-\gamma'_j)i^j} \bmod p \text{ for } i = 1, 2, \ldots, n$$

and (9)/(10) yields

$$C^{R-R'} = g^{\gamma_0-\gamma'_0} h^{\tau-\tau'} \bmod p.$$

Note that R and R' are different integers in Z_q and q is a prime and so $(R - R')^{-1} \bmod q$ exists. Therefore,

$$D_i(c_i) = g^{\sum_{j=0}^{t-1}(\gamma_j-\gamma'_j)/(R-R')i^j} \bmod p \text{ for } i = 1, 2, \ldots, n$$
$$C = g^{(\gamma_0-\gamma'_0)/(R-R')} h^{(\tau-\tau')/(R-R')} \bmod p$$

and thus each $\log_g D_i(c_i)$ is a share generated by polynomial $f(x) = \sum_{j=0}^{t-1}((\gamma_j - \gamma'_j)/(R - R'))x^j$ where discrete logarithm of the shared secret, $(\gamma_0 - \gamma'_0)/(R - R') \bmod N$, is committed to in C. □

4.3 Further Efficiency Improvement by Batch Verification

High efficiency for the dealer is very obvious in our new PVSS scheme. His only exponentiation computation in his proof of validity of his secret sharing includes encryption of k_is, namely n instances of ElGamal encryption, which costs $2n$ exponentiations. It is a great advantage over the existing PVSS schemes as to be detailed in Section 5. However, the computational cost for a verifier is not so efficient. Verification of Equations (1) and (2) for $i = 1, 2, \ldots, n$ costs $5n$ exponentiations, each with an exponent in Z_q. Verification of the two equations can be batched by a verifier using the idea in [3] as follows.

1. n integers $t_1, t_2, \ldots, t_n$ are randomly chosen by the verifier from Z_{2^L} where L is a security parameter such that $2^L < q$.
2. He verifies

$$(\prod_{i=1}^n a_i^{t_i})^R \prod_{i=1}^n a'^{t_i}_i = g^{\sum_{i=1}^n R_i t_i} \bmod p \quad (11)$$
$$(\prod_{i=1}^n b_i^{t_i})^R \prod_{i=1}^n b'^{t_i}_i = g^{\sum_{i=1}^n S_i t_i} \prod_{i=1}^n y_i^{R_i t_i} \bmod p. \quad (12)$$

This batch verification is a straightforward application of the principle in Theorem 4. Theorem 4 guarantees that if (11) and (12) are satisfied with a probability larger than 2^{-L} then (1) and (2) are satisfied. As explained in [3], the principle of batch verification is employment of small exponents. Bellare *et al* notice that the exponentiation computations in cryptographic operations usually employ very large exponents (hundreds of bits long) and sometimes the exponents are not necessary to be so large. Actually in many practical applications the exponents can be much smaller (e.g. scores of bits long) but still large enough to guarantee very strong security (e.g. to control the probability of failure under one out of

billions). So they employ many small exponents in verification of a batch of equations and estimate the computational cost in terms of the number of separate exponentiations with full-length exponents. More precisely, they set the computational cost of an exponentiation with a full-length exponent as the basic unit in efficiency analysis and estimate how many basic units cost the same as their operations[3]. In this way, they can clearly show advantages of batch verification in computational efficiency. More precisely, in our batch verification, 2^L can be much smaller than the integers in Z_q to improve efficiency, while correctness and soundness of the verification can still be guaranteed except for a probability of 2^{-L}.

Theorem 4. *Suppose $H, y_1, y_2, \ldots, y_n$ are in G_q, $t_1, t_2, \ldots, t_n$ are randomly chosen from $\{0, 1, \ldots, 2^L - 1\}$. If $\prod_{i=1}^n y_i^{t_i} = H^{\sum_{i=1}^n x_i t_i} \bmod p$ with a probability larger than 2^{-L}, then $y_i = H^{x_i} \bmod p$ for $i = 1, 2, \ldots, n$.*

Proof: $\prod_{i=1}^n y_i^{t_i} = H^{\sum_{i=1}^n x_i t_i} \bmod p$ with a probability larger than 2^{-L} implies that for any given integer v in $\{1, 2, \ldots, n\}$ there must exist integers $t_1, t_2, \ldots, t_n \in \{0, 1, \ldots, 2^L - 1\}$ and $t'_v \in \{0, 1, \ldots, 2^L - 1\}$ such that

$$\prod_{i=1}^n y_i^{t_i} = H^{\sum_{i=1}^n x_i t_i} \bmod p \tag{13}$$

$$(\prod_{i=1}^{v-1} y_i^{t_i}) y_v^{t'_v} \prod_{i=v+1}^n y_i^{t_i} = H^{(\sum_{i=1}^{v-1} x_i t_i) + x_v t'_v + \sum_{i=v+1}^n x_i t_i} \bmod p. \tag{14}$$

Otherwise, for any $(t_1, t_2, \ldots, t_{v-1}, t_{v+1}, \ldots, t_n)$ in $\{0, 1, \ldots, 2^L - 1\}^{n-1}$, there is at most one t_v in $\{0, 1, \ldots, 2^L - 1\}$ to satisfy $\prod_{i=1}^n y_i^{t_i} = H^{\sum_{i=1}^n x_i t_i} \bmod p$, which implies that among the 2^{nL} possible choices for $\{t_1, t_2, \ldots, t_n\}$ there are at most $2^{(n-1)L}$ choices to satisfy $\prod_{i=1}^n y_i^{t_i} = H^{\sum_{i=1}^n x_i t_i} \bmod p$ and leads to a contradiction to the assumption that $\prod_{i=1}^n y_i^{t_i} = H^{\sum_{i=1}^n x_i t_i} \bmod p$ with a probability larger than 2^{-L}.

(13)/(14) yields

$$y_v^{t_v - \hat{t}_v} = H^{(t_v - \hat{t}_v) x_v} \bmod p.$$

Note that t_v and $\hat{t}_v$ are L-bit integers and $2^L < q$. So $(t_v - \hat{t}_v)^{-1} \bmod q$ exists and thus

$$y_v = H^{x_v} \bmod p.$$

Therefore, $y_i = H^{x_i}$ for $i = 1, 2, \ldots, n$ as v can be any integer in $\{1, 2, \ldots, n\}$. □

With this batch optimisation, computational efficiency of a verifier is greatly improved. For a verifier, the total computational cost includes two full-length exponentiation and four instances of computation of product of n exponentiations with L-bit exponents and one instance of computation of product of n exponentiations with $\log_2 q$-bit exponents. According to [2], computing each of the four instances of product of exponentiations with L-bit exponents costs about $2^{W-1}(n + 1) + L + nL/(W + 1)$ multiplications and computing the product of n exponentiations with $\log_2 q$-bit exponents costs about

[3] Namely, multiple exponentiations with small exponents are counted as one exponentiation with a full-length exponent, which has the same cost.

$2^{W-1}(n+1) + |q| + n|q|/(W+1)$ multiplications where $|q|$ is the bit-length of q and W is a parameter in the W-bit-sliding-window exponentiation method and is normally set as 3. When standard W-bit-sliding-window exponentiation method is employed, an exponentiation with a full-length exponent in Z_q costs $2^{W-1} + |q| + |q|/(W+1)$ multiplications. So the computational cost of a verifier is approximately equal to

$$(4(2^{W-1}(n+1) + L + nL/(W+1)) + 2^{W-1}(n+1) + |q| + n|q|/(W+1)) \\ /(2^{W-1} + |q| + |q|/(W+1)) + 2$$

full-length exponentiations. When $L = 40$, 2^{-L} is smaller than one out of one trillion and thus negligible in any practical application. In this case, when $W = 3$ and $|q|$=1024, the computational cost of a verifier is very low.

5 Conclusion

As stated before, the new PVSS scheme needs no additional condition or assumption as it only needs the most basic assumptions absolutely needed in any PVSS scheme. The new PVSS scheme has advantages over the existing PVSS schemes in both security and efficiency. The extension of applicability in Section 4 shows that it can be widely applied to many distributed secure applications.

References

1. Adida, B., Wikström, D.: How to shuffle in public. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 555–574. Springer, Heidelberg (2007)
2. Avanzi, R., Cohen, H., Doche, C., Frey, G., Lange, T., Nguyen, K., Vercauteren, F.: Handbook of Elliptic and Hyperelliptic Curve Cryptography. In: HEHCC (2005)
3. Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998)
4. Boudot, F., Traoré, J.: Efficient publicly verifiable secret sharing schemes with fast or delayed recovery. In: Varadharajan, V., Mu, Y. (eds.) ICICS 1999. LNCS, vol. 1726, pp. 87–102. Springer, Heidelberg (1999)
5. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
6. Chandran, N., Ostrovsky, R., Skeith III, W.E.: Public-key encryption with efficient amortized updates. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 17–35. Springer, Heidelberg (2010)
7. Damgård, I., Thorbek, R.: Non-interactive proofs for integer multiplication. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 412–429. Springer, Heidelberg (2007)
8. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: FOCS 1987, pp. 427–437 (1987)
9. Fouque, P., Poupard, G., Stern, J.: Sharing decryption in the context of voting or lotteries. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 90–104. Springer, Heidelberg (2001)

10. Fujisaki, E., Okamoto, T.: A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 32–46. Springer, Heidelberg (1998)
11. Ge, H., Tate, S.: A direct anonymous attestation scheme for embedded devices. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 16–30. Springer, Heidelberg (2007)
12. Groth, J.: Non-interactive zero-knowledge arguments for voting. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 467–482. Springer, Heidelberg (2005)
13. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutylowski, M., Adida, B. (eds.) Towards Trustworthy Elections. LNCS, vol. 6000, pp. 37–63. Springer, Heidelberg (2010)
14. Kiayias, A., Yung, M.: Tree-homomorphic encryption and scalable hierarchical secret-ballot elections. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 257–271. Springer, Heidelberg (2010)
15. Küpçü, A., Lysyanskaya, A.: Optimistic fair exchange with multiple arbiters. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 488–507. Springer, Heidelberg (2010)
16. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
17. Peng, K., Bao, F.: Efficient publicly verifiable secret sharing with correctness, soundness and ZK privacy. In: Youm, H.Y., Yung, M. (eds.) WISA 2009. LNCS, vol. 5932, pp. 118–132. Springer, Heidelberg (2009)
18. Peng, K.: Verifiable secret sharing with comprehensive and efficient public verification. In: Li, Y. (ed.) DBSec 2011. LNCS, vol. 6818, pp. 217–230. Springer, Heidelberg (2011)
19. Peng, K.: Impracticality of efficient PVSS in real life security standard (Poster). In: Parampalli, U., Hawkes, P. (eds.) ACISP 2011. LNCS, vol. 6812, pp. 451–455. Springer, Heidelberg (2011)
20. Saxenaa, N., Tsudikb, G., Yic, J.: Threshold cryptography in p2p and manets: The case of access control. In: Computer Networks, vol. 51(12), pp. 3632–3649 (2007)
21. Schoenmakers, B.: A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 148–164. Springer, Heidelberg (1999)
22. Shamir, A.: How to share a secret. Communication of the ACM 22(11), 612–613 (1979)
23. Stadler, M.: Publicly verifiable secret sharing. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 190–199. Springer, Heidelberg (1996)