Verifiable Secret Sharing with Comprehensive and Efficient Public Verification

Kun Peng

Verifiable Secret Sharing With Comprehensive And Efficient Public Verification

Kun Peng

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

Abstract. VSS (verifiable secret sharing) is an important security protection tool in distributed systems. When VSS is employed in publicly verifiable applications, it needs to achieve public verifiability and be upgraded to PVSS (publicly verifiable secret sharing). Besides the two basic security properties, bindingness and hidingness, PVSS concentrates on public verifiability of validity all the operations in VSS so that there is no doubt about any operation and any dispute can be publicly solved. The existing PVSS schemes achieve security and public verifiability at a high cost. Moreover, their public verification operations are not defined and specified comprehensively and in complete details. In addition, most of them are vulnerable to an attack called simple plaintext attack. To overcome those drawbacks in PVSS, a new PVSS protocol is proposed in this paper. It defines public verifiability of VSS in a comprehensive and formal security model, which describes every verification operation in details and can publicly solve any dispute. All the public verification operations are efficiently implemented in the new PVSS protocol, which is more efficient than the existing PVSS schemes. It prevents simple plaintext attack in an efficient way.

1 Introduction

The first threshold secret sharing technique is Shamir's t-out-of-n secret sharing [14]. It is one of the most basic tools in distributed computing systems. A dealer holds a secret and shares it among n distributed share holders. Any t distributed share holders can put their shares together to reconstruct the secret, while no information about the secret is obtained if the number of available distributed shares is less than t. In practical applications, sometimes the share holders do not trust the dealer and want to verify that their shares are valid such that any t of them reconstruct the same secret. So VSS [4, 12, 8, 6, 12, 3, 1, 5, 9–11] is proposed, in which there is a verification mechanism for each share holder to verify validity of his share. In VSS, the share verification mechanism cannot reveal any information about the secret or any of its shares. In [13], this property is called hidingness, while in other schemes it has other names like zero knowledge, indistinguishability, confidentiality and privacy. We inherit the name hidingness and define it in a simulation-based formal model, which is popular in formal analysis

of privacy. Another important security property in VSS is bindingness, which is called soundness or consistency in some other places. A VSS protocol is binding if its verification mechanism guarantees that each share is generated from the same secret.

In some applications of VSS, validity of sharing and reconstruction of the secret need to be publicly verified or checked. So PVSS is designed. The existing PVSS schemes [15, 2, 7, 13, 11] have some drawbacks. Firstly, they are complex and inefficient, especially in computation. Moreover, public verification operations are not comprehensively defined and specified in them. Most of them only focus on proof of validity of encryption in terms of the shares, while other public verification operations like public dispute solution and public verification of reconstruction are not specified in details. In addition, most of them [15, 2, 13][1] publish a deterministic commitment of the secret and so are vulnerable to an attack called simple plaintext attack detailed in Section 2. The attack prevents application of the PVSS schemes in some environments. Although the attack can be prevented by replacing their commitment algorithms with probabilistic commitment algorithms, the countermeasure leads to more complex commitment, proof and verification operations, so further deteriorates efficiency of the PVSS schemes.

In this paper, a new PVSS protocol is designed. It achieves comprehensive public verifiability defined in complete details and in a formal security model. It is much more efficient than the existing PVSS schemes in both computation and communication. It is inherently invulnerable to the simple plaintext attack and does not need any additional cost to prevent it. The rest of this paper is arranged as follows. PVSS and its security properties, especially public verifiability, are formally and comprehensively defined and modeled in Section 2. The new PVSS protocol is proposed in Section 3. It is analysed in security and efficiency and compared with the existing PVSS schemes in Section 4. The paper is concluded in Section 5.

2 Security Model

In Shamir's t-out-of-n secret sharing [14], a dealer A uses the following procedure to share a secret s among n distributed share holders P_1, P_2, ..., P_n.

1. A builds a polynomial $f(x) = \sum_{j=0}^{t-1} a_j x^j$ where $a_0 = s$ and a_i for $j = 1, 2, \ldots, t-1$ are random integers.
2. A sends $s_i = f(i)$ as a share to P_i for $i = 1, 2, \ldots, n$.
3. Any t share holders can reconstruct the secret: $s = \sum_{i \in V} s_i w_i$ where $w_i = \prod_{j \in V, j \neq i} \frac{-j}{j-i}$ and V is the set containing the indexes of the t shares. Any group of less than t share holders get no information about the secret.

[1] A method is mentioned in [13] to prevent simple plaintext attack. However, it needs additional operations and approximately doubles the computational cost. In comparison with our much more efficient countermeasure against the attack, it is not good enough.

If the dealer is not trusted and the share holders want to verify validity of the shares, VSS is needed, which must satisfy the following two properties.

- Bindingness: there is a unique secret such that if the i^{th} share passes the validity verification, it is the i^{th} share of the unique secret.
- Hidingness: the verification mechanism does not reveal any information about the secret.

VSS and its security properties are formally defined as follows.

Definition 1 *(VSS) In a t-out-of-n VSS protocol, there exist explicitly or implicitly committed (to be detailed in Definition 2) integers $a_0, a_1, \ldots, a_{t-1}$ such that any share holder P_i can verify $s_i = \sum_{j=0}^{t-1} a_j i^j$.*

Definition 2 *An integer is explicitly committed to if its commitment is published. An integer is implicitly committed to if its unique existence can be proved such that successful verification of the proof with an overwhelmingly large probability guarantees that it is uniquely determined by the proof.*

Definition 3 *(Bindingness) In a t-out-of-n binding VSS protocol, if P_i's verification of validity of s_i is passed with a non-negligible probability, it is guaranteed that $s_i = \sum_{j=0}^{t-1} a_j i^j$ where integers $a_0, a_1, \ldots, a_{t-1}$ are publicly committed to or extracted from the dealer's proof as defined in Definition 2.*

Definition 4 *(Hidingness) A VSS protocol is hiding if the information revealed in its verification mechanism can be simulated without any difference by a party without any knowledge of the secret or any of its share.*

In applications of secret sharing requiring public verifiability, apart from the normal functionality of VSS, the following publicly verifiable operations are needed to publicly detect and handle various misbehaviors.

- Share distribution must be public. Namely, the dealer must publicly show that it really send a secret share to every share holder.
- Validity of shares should be publicly verified against a public commitment of the secret. If the public verification is passed, any share holder's knowledge of a valid share can be publicly recognised and he cannot complain of receiving an invalid share later.
- When the public verification of a share fails, there must be a public mechanism to detect whether the corresponding share holder or the dealer is cheating.
- When the secret is reconstructed, it can be publicly verified to be correct against the public commitment of the secret.
- When secret reconstruction fails, it can be publicly detected which share holder provides an invalid share and should be responsible.

These five operations are called public share distribution, public verification of share validity against public commitment, public solution to dispute in sharing, public verification of reconstruction and public detection of invalid share. More formally, in applications requiring public verifiability, PVSS is needed and Definitions 1, 3 and 4 should be extended to Definitions 5, 6 and 7 respectively.

Definition 5 *In a t-out-of-n PVSS protocol, encryptions of the shares are published. Moreover, there is a public commitment of the secret denoted as C. The following public proofs and verifications are necessary.*

- *(Public verification of share validity against public commitment) Any share s_i can be publicly verified through a proof and verification operation*

$$A\ (a_0, a_1, \ldots, a_{t-1}, \ldots) \longrightarrow \alpha$$
$$P_i\ (s_i, \ldots) \longrightarrow \beta_i$$

  where $P\ (\theta) \longrightarrow \gamma$ means that a party P with secret input θ outputs public information γ and "..." means that additional secret information may be used. The verification returns a public result

$$T(\alpha, \beta_i) = \begin{cases} TRUE \implies s_i \text{ is valid;} \\ FALSE \implies s_i \text{ is invalid.} \end{cases} \quad (1)$$

  There exist explicitly or implicitly committed (as defined in Definition 2) integers $a_0, a_1, \ldots, a_{t-1}$ such that if the verification result is $TRUE$ for any s_i it is guaranteed $s_i = \sum_{j=0}^{t-1} a_j i^j$ where a_0 is the secret committed in C.
- *(Public solution to dispute in sharing) When (1) returns $FALSE$, P_i has to publicly prove his honesty through a proof*

$$Proof_{(sk_i)}(C, E_i, \alpha, \beta_i)$$

  where sk_i denotes P_i's private key, E_i denotes the public encryption of P_i's share and $Proof_{(\tau)}(\delta)$ denote a proof using secret information τ against public information δ. If the proof is successful, the dealer is cheating; otherwise P_i is cheating.
- *(Public verification of reconstruction) The reconstructed secret ς, which includes s and perhaps other reconstruction results, can be publicly verified against C in*

$$Test\ (\varsigma, C, \alpha) \quad (2)$$

  If the test is successful, s is publicly accepted as the secret committed in C. Otherwise, ς is invalid and some share used to reconstruct s must be invalid.
- *(Public detection of invalid share) When (2) fails, each share holder P_i having participated in the reconstruction proves validity of the share he provides through a proof*

$$Proof_{(sk_i)}(E_i, \alpha, \beta_i)$$

  If the proof is successful, P_i is honest; otherwise P_i is cheating.

Definition 6 *(Bindingness in PVSS) If (1) returns $TRUE$ with a non-negligible probability, it is publicly guaranteed that $s_i = \sum_{j=0}^{t-1} a_j i^j$ where integers $a_0, a_1, \ldots, a_{t-1}$ are publicly committed to or extracted from the dealer's proof as defined in Definition 2.*

Definition 7 *(Hidingness in PVSS) Before any share is published, the published information can be simulated in a indistinguishable way by a party without any knowledge of the secret or any of its share. In PVSS, commitment of the secret is usually published. So a special attack against privacy defined in Definition 8 cannot be ignored.*

Definition 8 *(Simple plaintext attack) Given a message s' and C, the public commitment of the shared secret, a polynomial attacker wants to test whether s' is committed in C and thus is the shared secret. The game can be more formally described as follows.*

1. *A randomly choose a bit b. If $b = 0$, he sets $s' = s$; if $b = 1$, he randomly chooses s' from Z_q.*
2. *A sends s' and C to the attacker.*
3. *The attacker has to guess b.*

If the probability that the polynomial attacker can correctly guess b is non-negligible, the simple plaintext attack is successful.

In most applications of PVSS (e.g. e-voting) the message space contains some readable plaintexts with sensible meaning in a not-very-large range. So the simple plaintext attack is harmful. For example, by repeating it in a brute-force attack, an attacker can find the secret. When the attacker has some additional information about the secret, the attack is especially effective. So the PVSS schemes vulnerable to this attack [15, 2, 13] have limited applications.

Besides higher formality and comprehensiveness and awareness of simple plaintext attack, our definition has a novelty: the integers used to generate the shares (denoted as $a_0, a_1, \ldots, a_{t-1}$ in our definition) are not necessary to be explicitly committed to. As will be illustrated in Section 3, our design of PVSS takes advantage of this novel property to improve efficiency. As unlike in the existing PVSS schemes it is not necessary to make explicit commitment to $a_0, a_1, \ldots, a_{t-1}$ and publish their commitments, efficiency in computation, communication and storage can be greatly improved. In our design, the dealer proves his knowledge of the integers used to generate the shares such that when his proof can succeed with a non-negligible probability $a_0, a_1, \ldots, a_{t-1}$ can be uniquely extracted from his proof. As shown in Section 3.1, this novel mechanism in general works well in VSS, which only enables the share holders to verify their own shares and usually does not have a compulsory need to publish an explicit commitment of the secret. As sometimes in applications of PVSS an explicit public commitment of the secret is needed (especially when an application employs the PVSS mechanism and then securely handles the shared secret in a publicly verifiable manner), an explicit commitment of the secret (denoted as C) is still included in Definition 5. However, in comparison with the public and explicit commitment to t or $2t$ integers in the existing PVSS schemes, our design only explicitly commits to two integers and is much more efficient.

3 The New PVSS Protocol

The new PVSS protocol is proposed in this section. Firstly, its main idea is introduced. Then a detailed description is present.

3.1 The Main Idea

The main idea of the new PVSS technique is to specify VSS efficiently and then make all the operations publicly verifiable. VSS is designed in a novel way. After a secret is shared among some share holders, the dealer shares an additional random secret among them if they require verification of their shares. Then shares of the two secrets are combined by a random linear relation. Validity of the combined shares can be verified through Lagrange Interpolation. Verification of validity of the original shares is passed if and only if the combined shares are valid. The sharing and verification operations are as follows.

1. Setting:
   A large prime q is chosen. It must be at least larger than any possible secret to share.
2. Sharing:
   A builds a polynomial $F(x) = \sum_{j=0}^{t-1} a_j x^j$ where $a_0 = s$ and a_j for $j = 1, 2, \ldots, t-1$ are random integers chosen from Z_q. A sends $s_i = F(i) \bmod q$ as a share to P_i for $i = 1, 2, \ldots, n$.
3. Verification:
   If verification of validity of the shares are required, the following proof and verification procedure is run.
   (a) A randomly chooses an additional secret k in Z_q.
   (b) A builds a polynomial $G(x) = \sum_{j=0}^{t-1} b_j x^j$ where $b_0 = k$ and b_j for $j = 1, 2, \ldots, t-1$ are random integers chosen from Z_q.
   (c) A sends $k_i = G(i) \bmod q$ to P_i for $i = 1, 2, \ldots, n$.
   (d) A random integer r is chosen in some way (to be detailed in Section 3.2) in Z_q as a public challenge to A.
   (e) A publishes $c_j = b_j + r a_j \bmod q$ for $j = 0, 1, \ldots, t-1$.
   (f) Any P_i can verify
   $$k_i + r s_i = \sum_{j=0}^{t-1} c_j i^j \bmod q \tag{3}$$
   if he doubts validity of his share. He accepts validity of s_i if and only if the equation holds.
4. Reconstruction
   A set with at least t share holders can reconstruct the secret: $s = \sum_{i \in V} s_i w_i \bmod q$ where $w_i = \prod_{j \in V, j \neq i} \frac{j}{j-i} \bmod q$ and V is the set containing the indexes of the t shares.

To achieve public verifiability, commitment of the secret and encryption of the shares should be published, so that the four proof and verification operations defined in Definition 5 can be publicly implemented against them. Especially, a much simpler and more efficient commitment mechanism is employed in our new design than in the existing PVSS schemes. The advantages of the new idea is that it can achieve high efficiency in both computation and communication.

3.2 Detailed Description

The basic operations of the new PVSS protocol are described in Figure 1, while its proof and verification operations to support public verification as required in Definition 5 are described in Figure 2. Note that in comparison with the existing PVSS schemes we employ much more efficient symmetric cipher to encrypt the shares when distributing them and much more efficient commitment operations when committing to the secret.

1. Public commitment
   The dealer publicly publishes a commitment of his secret.
   (a) p and q are large primes such that $p = 2q + 1$. G is the cyclic subgroup of Z_p^* with order q. g and h are two generators of G such that $\log_g h$ is secret.
   (b) A has a secret s in Z_q to share. He randomly chooses an additional secret k in Z_q.
   (c) A publishes $u = g^s h^x \bmod p$ and $v = g^k h^y \bmod p$ where x and y are randomly chosen from Z_q.
2. Public share distribution
   The dealer encrypts the shares and publicly sends them to the share holders. Unlike the existing PVSS schemes, which employ costly asymmetric encryption algorithms, our new solution can employ a much more efficient symmetric encryption algorithm to encrypt the shares. However, for fairness in comparison with the existing PVSS schemes, we include distribution of the symmetric keys through asymmetric cipher in our share distribution procedure. The dealer publicly distributes the shares of secret s as follows.
   (a) A builds a polynomial $G(x) = \sum_{j=0}^{t-1} b_j x^j$ where $b_0 = k$ and b_j for $j = 1, 2, \ldots, t-1$ are random integers chosen from Z_q.
   (b) A calculates $k_i = G(i) \bmod q$ and publishes $d_i = E_i(k_i)$ for each P_i where $E_i()$ denotes encryption using P_i's public key and an asymmetric encryption algorithm (e.g. RSA or ElGamal).
   (c) A builds a polynomial $F(x) = \sum_{j=0}^{t-1} a_j x^j$ where $a_0 = s$ and a_j for $j = 1, 2, \ldots, t-1$ are random integers chosen from Z_q.
   (d) A calculates $s_i = F(i) \bmod q$ and publishes $e_i = E_{k_i}(s_i)$ for each P_i where $E_{k_i}()$ denotes symmetric encryption using session key k_i and an symmetric encryption algorithm (e.g. AES).
3. Reconstruction
   If at least t share holders submit their shares s_i, the secret s is reconstructed:

$$s = \sum_{i \in V} s_i w_i \bmod q \text{ where } w_i = \prod_{j \in V, j \neq i} \frac{j}{j-i} \bmod q$$

   and V is the set containing the indexes of the t shares.

Fig. 1. Basic Operations of PVSS

– Public verification of share validity against public commitment
  1. $r = H(d_1, d_2, \ldots, d_n, e_1, e_2, \ldots, e_n)$ is generated where $H()$ is a one-way and collision-resistent hash function with an image domain Z_q.
  2. An P_i requesting public verification of his share decrypts d_i and obtains k_i, which is then used to decrypt e_i into s_i.
  3. P_i publishes $h_i = H(k_i + rs_i)$.
  4. A publishes $c_j = b_j + ra_j \bmod q$ for $j = 0, 1, \ldots, t-1$
     $z = y + rx \bmod q$.
  5. P_i publishes $z_i = k_i + rs_i \bmod q$.
  6. It is publicly verified
$$g^{c_0} h^z = u^r v \bmod p \quad (4)$$
$$z_i = \sum_{j=0}^{t-1} c_j i^j \bmod q \quad (5)$$
$$h_i = H(z_i) \quad (6)$$

  Validity of s_i is accepted if and only if all the three equations (4) , (5) and (6) are satisfied. Their satisfaction publicly guarantees that P_i knows the i^{th} share of the secret committed in u. If (4) fails, the dealer is dishonest and must have cheated. If (6) fails, P_i is dishonest and must have cheated. If (5) fails, there is a dispute between the dealer and P_i about validity of s_i, which can be solved in the next proof and verification operation.
– Public solution to dispute in sharing
  If (5) fails, P_i has to publish k_i and s_i. Any one can verify
$$k_i + rs_i \neq \sum_{j=0}^{t-1} c_j i^j \bmod q$$
$$e_i = E_{k_i}(s_i)$$
$$d_i = E_i(k_i)$$

  If they are satisfied, P_i is honest and the dealer must have given him an invalid share. Otherwise P_i is dishonest and must have tampered with his share.
– Public verification of reconstruction
  1. k is reconstructed: $k = \sum_{i \in V} k_i w_i \bmod q$ where $w_i = \prod_{j \in V, j \neq i} \frac{j}{j-i} \bmod q$
  2. It is publicly verified
$$(u/g^s)^r (v/g^k) = h^z \bmod p. \quad (7)$$

  s passes the verification for its validity only if (7) is satisfied. Note that if necessary a new r in Z_q can be randomly chosen (or generated by a hash function) and a new z can be calculated as $z = y + rx \bmod q$.
– Public detection of invalid share
  If (7) is not satisfied, the reconstructed s is invalid and some invalid share must have been used in the reconstruction. So for each s_i used in the reconstruction
$$k_i + rs_i = \sum_{j=0}^{t-1} c_j i^j \bmod q$$
$$e_i = E_{k_i}(s_i)$$
$$d_i = E_i(k_i)$$

  are publicly verified. If any of the three equation fails for an s_i, the corresponding P_i must have given an invalid share and should be responsible for failure of the reconstruction.

Fig. 2. Proof and Verification Operations in PVSS

4 Analysis and Comparison

The four proof and verification operations in Figure 2 satisfy the security requirements in Definition 5, Definition 6 and Definition 7. Firstly, Theorem 1 illustrates that the "public verification of share validity against public commitment" operation implements public verification of validity of the shares. Secondly, it is straightforward that the "public solution to dispute in sharing" operation can publicly solve any dispute between the dealer and any share holder about validity of any share. Thirdly, Theorem 2 illustrates that the "public verification of reconstruction" operation publicly guarantees that the reconstructed secret s is committed in u. Fourthly, it is straightforward that the "public detection of invalid share" operation can publicly detect any invalid share provided by any share holder. So the new PVSS protocol is binding and improves public verifiability. Hidingness of the new PVSS protocol is proved in Theorem 3, which guarantees that the secret and its shares are not revealed and especially formally guarantees that the new PVSS protocol is invulnerable to the simple plaintext attack.

Theorem 1. *The new PVSS protocol can publicly guarantee validity of the shares against a public commitment of the secret. More precisely, if (4), (5) and (6) are satisfied with a probability larger than* $1/q$, s_i *is publicly guaranteed to be the* i^{th} *share of the secret explicitly committed in* u *and generated by a polynomial determined by a set of implicitly committed integers* $a_0, a_1, \ldots, a_{t-1}$ *as defined in Definition 5 and Definition 6.*

Proof: As (4), (5) and (6) are satisfied with a probability larger than $1/q$, there must exist two different integers r and r' in Z_q such that A and P_i can provide $z, c_0, c_1, \ldots, c_{t-1}, z_i$ and $z', c'_0, c'_1, \ldots, c'_{t-1}, z'_i$ respectively to satisfy

$$z_i = \sum_{j=0}^{t-1} c_j i^j \bmod q \tag{8}$$

$$z'_i = \sum_{j=0}^{t-1} c'_j i^j \bmod q \tag{9}$$

$$g^{c_0} h^z = u^r v \bmod p \tag{10}$$

$$g^{c'_0} h^{z'} = u^{r'} v \bmod p. \tag{11}$$

Otherwise, there is at most one r in Z_q for A and P_i to provide $z, c_0, c_1, \ldots, c_{t-1}$ and z_i to satisfy (4), (5) and (6) and the probability that they are satisfied is no larger than $1/q$, which is a contradiction.

Integers s_i, k_i, $a_0, a_1, \ldots, a_{t-1}$ and $b_0, b_1, \ldots, b_{t-1}$ to satisfy

$$z_i = r s_i + k_i \bmod q \tag{12}$$

$$z'_i = r' s_i + k_i \bmod q \tag{13}$$

$$c_j = r a_j + b_j \bmod q \text{ for } j = 0, 1, \ldots, t-1 \tag{14}$$

$$c'_j = r' a_j + b_j \bmod q \text{ for } j = 0, 1, \ldots, t-1 \tag{15}$$

can be calculated as

$$\begin{pmatrix} s_i \\ k_i \end{pmatrix} = \begin{pmatrix} r & 1 \\ r' & 1 \end{pmatrix}^{-1} \begin{pmatrix} z_i \\ z'_i \end{pmatrix}$$

$$\begin{pmatrix} a_j \\ b_j \end{pmatrix} = \begin{pmatrix} r & 1 \\ r' & 1 \end{pmatrix}^{-1} \begin{pmatrix} c_j \\ c'_j \end{pmatrix} \text{ for } j = 0, 1, \ldots, t-1.$$

where the calculations in the matrices are performed modulo q. Note that (8), (12) and (14) imply

$$k_i + rs_i = \sum_{j=0}^{t-1} (ra_j + b_j) i^j \bmod q; \tag{16}$$

(9), (13) and (15) imply

$$k_i + r's_i = \sum_{j=0}^{t-1} (r'a_j + b_j) i^j \bmod q; \tag{17}$$

10) and (14) imply

$$g^{ra_0 + b_0} h^z = u^r v \bmod p; \tag{18}$$

and 11) and (15) imply

$$g^{r'a_0 + b_0} h^{z'} = u^{r'} v \bmod p. \tag{19}$$

Moreover, (16)-(17) yields

$$(r - r')s_i = \sum_{j=0}^{t-1} (r - r') a_j i^j \bmod q; \tag{20}$$

while (18)/(19) yields

$$g^{(r-r')a_0} h^{z-z'} = u^{r-r'} \bmod p. \tag{21}$$

Also note that r and r' are different integers in Z_q and q is a prime and thus $r - r' \neq 0 \bmod q$. So (20) implies

$$s_i = \sum_{j=0}^{t-1} a_j i^j \bmod q$$

and (21) implies

$$g^{a_0} h^{(z-z')/(r-r')} = u \bmod p.$$

Therefore, s_i is the i^{th} share of a_0, which is publicly committed in u. □

Theorem 2. *If (7) is satisfied with a probability larger than $1/q$, the reconstructed secret s is committed to by the dealer in u.*

Proof: As (7) is satisfied with a probability larger than $1/q$, there must exist two different integers in Z_q, r and r', such that given them A can return z and z' respectively to satisfy

$$(u/g^s)^r (v/g^k) = h^z \bmod p \tag{22}$$

$$(u/g^s)^{r'} (v/g^k) = h^{z'} \bmod p \tag{23}$$

Otherwise, there is at most one r in Z_q for A to produce a z to satisfy (7) and the probability that (7) is satisfied is no larger than $1/q$, which is a contradiction.

(22)/(23) yields

$$(u/g^s)^{r-r'} = h^{z-z'} \bmod p \tag{24}$$

If the dealer commits to an integer other than s in u, he must know integers s' and x' such that

$$u = g^{s'} h^{x'} \bmod p \tag{25}$$

and $s \neq s' \bmod q$. Note that

(24) and (25) imply

$$g^{(s'-s)(r-r')} = h^{z-z'-x'(r-r')} \bmod p.$$

Also note that r and r' are different integers in Z_q and q is a prime and thus $r - r' \neq 0 \bmod q$. So

$$\log_g h = (z - z' - x'(r - r'))/((s' - s)(r - r')) \bmod q$$

can be calculated in polynomial time, which is contradictory to the assumption that $\log_g h$ is secret and the DL problem is hard. Therefore, the reconstructed secret s is committed in u. □

Theorem 3. *The new PVSS protocol is hiding.*

Proof: We firstly show that the basic operations and one of the proof and verifications operations is hiding and then illustrate that the other three proof and verification operations dose not compromise hidingness although they need to publish some shares. The basic operations and one of the proof and verification operations "public verification of share validity against public commitment" have a transcript $u, v, d_1, d_2, \ldots, d_n, e_1, e_2, \ldots, e_n, r, c_0, c_1, \ldots, c_{t-1}, z, z_1, z_2, \ldots, z_n, h_1, h_2, \ldots, h_n$. Security of the employed encryption algorithm guarantees that it is difficult to extract the shares from $d_1, d_2, \ldots, d_n, e_1, e_2, \ldots, e_n$. The other variables in the transcript can be simulated by a party without any knowledge of any secret as follows.

1. He randomly chooses integers $r, z, c_0, c_1, \ldots, c_{t-1}$ from Z_q.
2. He calculates $z_i = \sum_{j=0}^{t-1} c_j i^j \bmod q$ and $h_i = H(z_i)$ for $i = 1, 2, \ldots, n$.
3. He randomly chooses integers s and x from Z_q and calculates $u = g^s h^x \bmod p$.
4. He calculates $k = c_0 - rs \bmod q$, $y = z - rx \bmod q$ and $v = g^k h^y \bmod p$.

In the random oracle model, distribution of r in the simulated transcript and in the real transcript are indistinguishable. Distribution of the simulated transcript and that of the real transcript of $u, v, c_0, c_1, \ldots, c_{t-1}, z, z_1, z_2, \ldots, z_n, h_1, h_2, \ldots, h_n$ are the same as follows.

– Each of $c_0, c_1, \ldots, c_{t-1}, z, z_1, z_2, \ldots, z_n$ is uniformly distributed in Z_q.
– Each of u and v is uniformly distributed in G.
– Each of $h_1, h_2, \ldots, h_n$ has a distribution $D\ \{h \mid h = H(Z),\ Z \text{ is uniformly distributed in } Z_q\}$ where $D\ \{\mu \mid \nu\}$ denote distribution of a variable μ determined by condition ν.
– The variables satisfy

$$g^{c_0} h^z = u^r v \bmod p$$
$$z_i = \sum_{j=0}^{t-1} c_j i^j \bmod q$$
$$h_i = H(z_i).$$

So no secret information is revealed from $u, v, r, c_0, c_1, \ldots, c_{t-1}, z, z_1, z_2, \ldots, z_n$, $h_1, h_2, \ldots, h_n$.

In "public solution to dispute in sharing", a share claimed to be invalid by its holder is published. If the share is really invalid, it is unrelated with the secret, so does not reveal any secret information. What if a malicious share holder takes this chance of disputing his share to reveal a valid share? Does such a revealing compromise hidingness of PVSS? Our answer is that if a share holder wants to reveal his share, he can do it anytime using any chance and the revealing cannot be prevented at all. So if a procedure is used by a malicious share holder as a chance to reveal his share, the procedure is not to blame. Actually, a basic assumption for any secret sharing technique to work is that each share holders will not reveal his share unless in the secret reconstruction phase. In "public verification of reconstruction", the secret has been reconstructed, so there is no secret information to be revealed. In "public detection of invalid share", the shares are published. Do the published valid shares violate privacy of PVSS? Our answer is that they do not as the PVSS protocol has passed the reconstruction phase when they are published. In the reconstruction phase the secret should be recovered and it is senseless to conceal its shares afterwards.

Therefore, none of the operations in the new PVSS protocol violates hidingness. □

The new PVSS protocol and the existing PVSS protocols are compared in Table 1. Neither the new PVSS protocol nor the existing PVSS protocols employs extensive communication, so communication is not compared although the new extended VSS is slightly more efficient in total communication including publishing encrypted shares and commitment values and transferring proof transcripts. We focus our comparison on computational cost, the bottleneck of PVSS. Our estimation of computational cost is comprehensive and takes into account the exponentiations needed in all the operations. When estimating computational cost of our new PVSS protocol, we have an observation: exponentiations with small bases and exponents like i^j is much less costly than an exponentiation usually used in asymmetric crypto, which has a base in a large cyclic group and a large exponent only limited by the order of the cyclic group. So, in efficiency analysis of threshold secret sharing (e.g. [9, 13]), an exponentiation used in Lagrange Interpolation is usually not treated like an exponentiation in the asymmetric

crypto operations, but counted as a small number of multiplications. Therefore, the number of all the modulo exponentiations in large cyclic groups (with bases and exponents usually hundreds of bits long) are counted in our estimation, while exponentiations with small bases and exponents (like i^j in Lagrange Interpolation where i and j are no larger than indexes of the share holders) are ignored. In [15], K is the time a proof and verification primitive with 1-bit challenge has to be repeated to guarantee soundness with a probability $1-2^{-K}$. For fairness of the comparison, when any implementation detail of the new PVSS protocol (e.g. choice of parameters or underlying algorithms) is not explicitly determined, it is assumed to be the same as that in most of the existing PVSS schemes. The four proof and verification operations defined in Definition 5 are denoted as PV1, PV2, PV3 and PV4. In PV1, PV2 and PV4, cost for a share is counted. Except for [7], the existing PVSS schemes employ deterministic commitment algorithms for public commitment of the secret. So they are vulnerable to the simple plaintext attack. As stated before, overcoming the vulnerability is costly and further deteriorates their efficiency. The comparison demonstrates that the new PVSS scheme achieves stronger security and higher efficiency than the existing PVSS schemes.

Table 1. Comparison of PVSS Schemes

scheme	encry--ption	commit-ment	decryption and recon--struction	PV1	PV2	PV3	PV4	simple plaintext attack
[7]	n	$2t$	n	$n+t$ +38	0	t	0	no
DL based [15]	$2n$	t	$2n$	$t+7K$	0	1	2	yes
e^{th} root based [15]	$2n$	t	$2n$	$t+6$	0	1	2	yes
[2]	n	t	n	$t+15$	0	1	1	yes
[13]	n	t	$2n$	$t+8$	0	t	2	yes
new PVSS	n	4	n	3	1	4	1	no

5 Conclusion

The new PVSS protocol improves efficiency and security of PVSS. It achieves comprehensive and complete public verifiability and is more efficient than the existing PVSS schemes. Moreover, it is inherently invulnerable to the simple plaintext attack.

References

1. C Benaloh. Secret sharing homomorphisms: keeping shares of a secret secret. In *CRYPTO '86*, pages 251–260.
2. F Boudot and J Traore. Efficient public verifiable secret sharing schemes with fast or delayed recovery. In *ICICS '99*, pages 87–102.
3. C Cachin, K Kursawe, A Lysyanskaya, and R Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In *ACM CCS '02*, pages 88–97.
4. D Chaum, Cl Crepeau, and I Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, STOC1988*, pages 11–19, 1988.
5. P Feldman. A practical scheme for non-interactive verifiable secret sharing. In *FOCS '87*, pages 427–437.
6. M Fitzi, J Garay, S Gollakota, C Rangan, and K Srinathan. Round-optimal and efficient verifiable secret sharing. In *TCC '06*, pages 329–342.
7. E Fujisaki and T Okamoto. A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In *EUROCRYPT '98*, pages 32–46.
8. R Gennaro and S Micali. Verifiable secret sharing as secure computation. In *EUROCRYPT '95*, pages 168–182.
9. T Pedersen. Distributed provers with applications to undeniable signatures. In *EUROCRYPT '91*, pages 221–242.
10. T Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91*, pages 129–140.
11. K Peng and F Bao. Efficient publicly verifiable secret sharing with correctness. In *WISA '09, LNCS5932*, pages 118–132.
12. T Rabin and M Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *ACM STOC '89*, pages 73–85.
13. B Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *CRYPTO '99*, pages 149–164.
14. A Shamir. How to share a secret. *Communication of the ACM*, 22(11):612–613, Nov 1979.
15. M Stadler. Publicly verifiable secret sharing. In *EUROCRYPT '96*, pages 190–199.