Provably Secure Proactive Secret Sharing Without the Adjacent Assumption

Zhe Xia[1], Bo Yang[2(✉)], Yanwei Zhou[2], Mingwu Zhang[3,4], Hua Shen[3], and Yi Mu[5]

[1] School of Computer Science and Technology, Wuhan University of Technology, Wuhan, China
xiazhe@whut.edu.cn
[2] School of Computer Science, Shaanxi Normal University, Xi'an, China
{byang,zyw}@snnu.edu.cn
[3] School of Computers, Hubei University of Technology, Wuhan, China
csmwzhang@gmail.com, cshshen@hbut.edu.cn
[4] State Key Laboratory of Cryptology, Beijing, China
[5] Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China
ymu.ieee@gmail.com

Abstract. In secret sharing (SS), the secret is shared among a number of parties so that only a quorum of these parties can recover the secret, but a smaller set of parties cannot learn any information about the secret. However, the traditional SS technique is insufficient to protect the secret with a long lifetime, because the adversary may gradually compromise enough parties to retrieve the secret over the long time. To solve this issue, proactive secret sharing (PSS) divides the lifetime of the secret into many short time periods and the parties jointly update their secret shares in each time period. The benefit is that if the adversary cannot break into enough parties in a single time period, her compromised shares will become obsolete after the shares being updated.

In the last two decades, many PSS schemes have been proposed and they are widely used in various security protocols. However, the majority of existing PSS schemes require the *adjacent assumption*, i.e. if a party is corrupted during an update phase, it is corrupted in both time periods adjacent to that update phase. Note that this assumption not only hinders the security model to capture the mobile adversary's abilities, but also prevents PSS schemes being used in many real-world applications. In this paper, we revisit the research of PSS, and our work contributes in the following aspects. Firstly, we discuss why some existing schemes (including Herzberg's PSS scheme) cannot maintain their security when the adjacent assumption is removed. Secondly, we use the polynomial truncation method to improve Herzberg's PSS scheme. To the best of our knowledge, our proposed scheme is the first provably secure PSS scheme without the adjacent assumption.

R. Steinfeld and T. H. Yuen (Eds.): ProvSec 2019, LNCS 11821, pp. 247–264, 2019.
https://doi.org/10.1007/978-3-030-31919-9_14

1 Introduction

The secret sharing (SS) technique, first introduced by Shamir [28] and Blakley [5], is an important building block in cryptology to protect secrecy and availability of sensitive information. The secret is divided into a number of shares and each share is held by an individual party. Therefore, if the adversary wants to learn or destroy the secret, she has to break into multiple parties. For example, in a (t, n)-threshold secret sharing, the secret is shared among n parties so that any t parties work together can recover the secret, but less than t parties cannot learn any information of the secret. And the adversary needs to compromise at least $n - t + 1$ parties if her purpose is to destroy the secret. However, the traditional SS technique is not suitable for some cases. For example, it might be insufficient to protect the secret with a long lifetime, e.g. crypto master keys, legal documents, medical records, etc. In these cases, the adversary may gradually compromise enough parties to learn the secret or destroy it, because she breaks into the parties in a monotonic fashion and she has a very long time to mount the attack.

To mitigate the above issue, proactive secret sharing (PSS) has been introduced in which the entire lifetime of the secret is divided into many short time periods and the parties jointly update the shares at the beginning of each time period with the original secret unchanged. The update includes a *share recovery* protocol and a *share refreshment* protocol. In the share recovery protocol, any lost or tampered share is recovered for the corresponding party without disclosing it to the other parties. In the share refreshment phase, the parties interactively compute new shares of the same secret and erase old shares. Because old shares and new ones are independent, if the adversary cannot break into enough parties before the update, any compromised share learned by the adversary will become obsolete after the update. In the case of a (t, n)-threshold PSS, the adversary has to compromise t parties in a single time period in order to learn the secret. This is opposed to compromising t parties over the entire lifetime in traditional SS schemes. For example, suppose some legal document needs to be protected for 10 years. If the shares are updated weekly, then the time slot for the adversary to break into t parties has been dramatically reduced from 10 years to 1 week.

The motivation of PSS is to protect the secret against the *mobile adversary* [23] who can compromise different parties at different time periods. Throughout the entire lifetime of the secret, the mobile adversary may corrupt all parties or break into some parties several times. But the requirement is that she can only compromise less than a quarom of parties in each time period. If a party is no longer corrupted by the mobile adversary, it will be "rebooted" into the safe state immediately.

Informally, proactive security refers to *secrecy* and *robustness* in the presence of the mobile adversary, where secrecy guarantees that the mobile adversary cannot learn any information about the secret in the entire lifetime of the secret, and robustness ensures that the secret can be correctly reconstructed in any time period even in the presence of some corrupted parties. Moreover, a PSS scheme is said to be *optimal resilient* if it is robust against any minority of cor-

rupted parties. Note that this threshold is the maximum number of corrupted parties allowed in SS schemes. In the literature, the threat model widely used in analysing PSS schemes requires the adjacent assumption, i.e. if a party is corrupted during an update phase, it is corrupted in both time periods adjacent to that update phase. In this paper, we investigate the necessity and implications of the adjacent assumption, and explore the design of provably secure PSS schemes without this assumption.

1.1 Related Works

The concept of mobile adversary was first introduced by Ostrovsky and Yung in [23]. The same paper also showed that if there exists pairwise secure communication channels and the parties can erase part of their memory, a lot of secure multiparty computation protocols (e.g. [3,10,26]) can be extended to withstand the mobile adversary. However, this idea only works theoretically, because the computation is done by secure distributed circuit evaluations and the communication costs are proportional to the size of circuits. For some specific problems, more efficient solutions are desired. Later, Canetti and Herzberg [9] introduced an efficient method to construct a distributed pseudorandom generator that can be maintained proactively. Canetti et al. [8] also demonstrated how to ensure authenticated and secret communication among parties that is robust against break-ins and key exposures.

Among the research of proactive security, PSS has attracted the most interests. Not only because it is a useful technique to protect secret with a long lifetime, but also it is an important building block for various security protocols, such as proactive threshold cryptosystems [19], proactive secure multiparty computation [2,30], key management in the ad hoc networks [18,31], and so on. In PSS, the secret is initially shared among the parties. The tricky part is how to jointly update the shares among the parties. For this task, three approaches have been introduced that achieve the optimal resilience property:

- **Herzberg's approach** [20]**:** before the update, the secret s is shared among the parties in a (t, n)-threshold fashion using a $t-1$ degree polynomial $f(x)$ such that $f(0) = s$. To update the shares, the parties jointly generate a random $t-1$ degree polynomial $\delta(x)$ with $\delta(0) = 0$. After the update, each party holds a new share of the $t-1$ degree polynomial $f'(x) = f(x) + \delta(x)$. Because $f'(0) = f(0) + \delta(0) = s$, the shares have been updated without changing the original secret s.
- **Frankel's approach** [12]**:** before the update, the secret is also shared among the parties in a (t, n)-threshold fashion. To update the shares, the parties first jointly transform the (t, n) polynomial sharing of the secret into an (n, n) additive sharing of the secret. To achieve optimal resilience, each share of the (n, n) additive sharing is further shared among the parties in the (t, n)-threshold fashion. Then, the parties jointly transform the (n, n) additive sharing of the secret back to a (t, n) polynomial sharing of the secret. Note that in both transformations, the secret is not revealed to any individual party, and after

the update, the polynomial used to share the secret is independent from the one before the update.
- **Rabin's approach** [25]**:** before the update, the secret is additively shared among the parties. To achieve optimal resilience, each of the old share is further shared among the parties in the (t, n)-threshold fashion. To update the shares, each party first shares her old share among the parties using another (n, n) additive sharing. In this process, each party will receive a share of the old share, called *fragement*, from every other party. Then, each party sums the received fragements, obtaining the new share of the secret. For optimal resilience, each party also needs to further share this new share among the parties in the (t, n)-threshold fashion. Now, the new shares form an independent (n, n) additive sharing of the original secret.

Based on the above PSS schemes, a lot of further investigations have been carried out in proactive security in the last two decades. For example, Stinson and Wei [29] have proposed an unconditionally secure PSS scheme using symmetric bivariate functions, in which both the secrecy and robustness properties are unconditionally protected. Canetti [7], followed by Frankel [14] and Almansa [1], have introduced the methods to extend PSS to withstand adaptive adversaries who can choose which parties to corrupt at any time during the run of the protocol. Cachin [6] and Zhou [32] have introduced PSS that is secure in the asynchronous communication model. Schultz et al. [27] have introduced a mobile PSS scheme that allows on-the-fly reconfiguration of the threshold, so that the scheme is able to accommodate more changes in the environment.

1.2 Our Contributions

In this paper, we revisit the research of provably secure and optimal resilient PSS, and we contribute in the following aspects:

- Firstly, although the adjacent assumption is widely used in existing PSS schemes, it not only hinders the security model to capture the mobile adversary's abilities, but also prevents PSS schemes from being used in many real-world applications. However, if this assumption is removed, we show that some existing schemes (including Herzberg's PSS scheme) will become insecure.
- Secondly, we use the polynomial truncation method to improve Herzberg's PSS scheme, resulting a provably secure PSS scheme without the adjacent assumption. To the best of our knowledge, it is the first PSS scheme satisfying this feature.

1.3 Organisation of the Paper

The rest of the paper is organised as follows: Sect. 2 outlines some preliminaries, including a new threat model without the adjacent assumption and some cryptographic building blocks. In Sect. 3, we show that the secrecy property in Herzberg's PSS scheme might be violated by the mobile adversary in our threat model. In Sect. 4, we use the polynomial truncation method to modify Herzberg's scheme, making it secure in our threat model. Finally, we conclude in Sect. 5.

2 Preliminaries

2.1 Models and Definitions

The Players. The players in our environment are n parties $P_1, P_2, \ldots, P_n$ and the mobile adversary $\mathcal{A}$. We assume that all these players can be modelled as probabilistic polynomial time (PPT) Turing machines [17]. Moreover, we assume that the system is synchronised, the parties can access to some common global clock, and each party has a local source of randomness. In this paper, we denote $n = 2t - 1$, where t is the threshold.

Time Periods. The entire lifetime of the secret is divided into many short time periods (e.g. a day, a week, etc.) which are determined by the common global clock. At the beginning of the first time period, there is a share distribution phase in which the secret is shared among the parties either by a trusted party or in a distributed fashion [16]. For all the other time periods, there is an update phase at the beginning of each time period. After the update, the lost or tampered shares are recovered and the parties hold new shares of the secret while the old shares are erased.

The Mobile Adversary. Following the description in [23], the mobile adversary can be envisioned as follows: it has $t - 1$ pebbles, and at the beginning of each time period, she places the pebbles on any $t - 1$ parties. If the pebble is placed on a party, this party is corrupted by the mobile adversary. Corrupting a party means learning this party's private information, changing its intended behaviour, disconnecting it, and so on. When the pebble is removed from a party, this party will be "rebooted" to the safe state at the beginning of the next time period, and her share will be jointly recovered by the parties. After each time period, the mobile adversary can move pebbles from a set of parties to a different set of parties. Therefore, the mobile adversary has more power than the ordinary adversary in traditional SS schemes, because the mobile adversary may corrupt all parties or break into some parties multiple times throughout the lifetime of the secret. However, it is assumed that the mobile adversary corrupts less than t parties in each time period.

The Communication Channel. We assume that all players are connected to a common authenticated broadcast channel $\mathcal{C}$, such that any message sent through $\mathcal{C}$ can be heard by the other players. The mobile adversary cannot modify messages send by an uncorrupted party through $\mathcal{C}$, nor she can prevent an uncorrupted party from receiving messages from $\mathcal{C}$. Moreover, we assume that there are pairwise secure communication channels among the parties, and the mobile adversary is unable to tamper or intercept the messages send through these secure channels. With these assumptions, we can focus our discussions on the proactive secret sharing schemes without considering the low level technical details. We note that both the authenticated broadcast channel and the pairwise secure channels can be implemented using standard techniques such as encryption and signature functions.

In the majority of existing PSS schemes (e.g. [1,12–15,20,21,27]), there is an assumption that if a party is corrupted during an update phase, it is corrupted during both time periods adjacent to that update phase. In comparison, our threat model does not require this assumption. We only assume that if a party is corrupted during an update phase, it is corrupted in the same time period but not in the preceeding time period. We show that this gives the adversary more power and such an adversary better mimics the mobile adversary. To simplify the description, considering the case that the entire lifetime of the secret has been divided into two time periods (as shown in Fig. 1). In the existing works, the adversary who corrupts $t-1$ parties during the update phase will corrupt the same parties throughtout the lifetime of the secret. In this case, the mobile adversary does not have more power than the ordinary adversary in traditional SS schemes. But in our threat model, the mobile adversary can corrupt some parties in time period 1 and then move to corrupt some other parties in time period 2. Therefore, the mobile adversary in our threat model has more power and our model better captures the ability of the mobile adversary. Moreover, the adjacent assumption will prevent the PSS schemes being used in many real-world applications. For example, PSS schemes were suggested to be used in Ad Hoc networks to safeguard the crypto keys in the distributed fashion [18,31]. But since the topology structure of the networks may change dynamically, and nodes may join or leave any time, the existing PSS schemes with the adjacent assumption are not suitable for these circumstances.

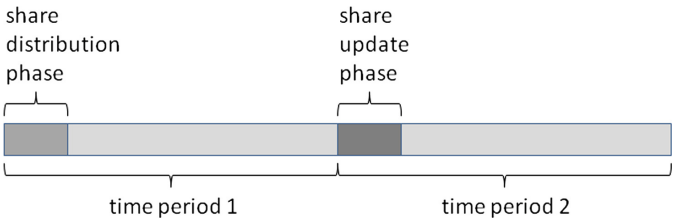

Fig. 1. A demonstration of the time periods

In order to provide rigorous security analysis for our proposed PSS scheme, we use the following security definitions:

Definition 1 (Robustness:) *A proactive secret sharing scheme is robust if in the presence of the mobile adversary, the secret can be correctly recovered in any time period throughout the entire lifetime of the secret.*

Definition 2 (Secrecy:) *A proactive secret sharing scheme is secret if after polynomially many updates, the mobile adversary still cannot learn any information of the secret.*

Definition 3 (Optimal resilience:) *A proactive secret sharing scheme is optimal resilient if it is robust against the mobile adversary who has the ability to corrupt any minority of the parties.*

2.2 Cryptographic Building Blocks

Shamir's Secret Sharing [28]**.** Denote p as a large prime such that $p > n$. In the rest of this paper, we assume that all computations are modulo p unless otherwise stated. To share the secret $s \in \mathbb{Z}_p$, the dealer first generates a polynomial $f(x) = a_0 + a_1 x + \cdots + a_{t-1}x^{t-1}$ over $\mathbb{Z}_p$ with degree $t-1$ such that $a_0 = s$. Then the dealer evaluates the polynomial $f(x)$ at different public and pre-defined values x_i for $i \in \{1, 2, \ldots, n\}$, and she sends the share $s_i = f(x_i)$ to the party P_i through the secure channel. If any t parties work together, they can recover the secret using polynomial interpolation as $s = \sum_{i=1}^{t} s_i \cdot L_i$, where $L_i = \prod_{j=1, j\neq i}^{t} x_j/(x_j - x_i)$ is the Lagrange coefficient. It is obvious that Shamir's SS is correct. To see why any $t-1$ colluding parties cannot learn any information of the secret, the $t-1$ points $(x_1, s_1), \ldots, (x_{t-1}, s_{t-1})$ are known by these parties. But for each possible value $s' \in \mathbb{Z}_p$, the point $(0, s')$ can be used to interpolate a unique polynomial, and the probability of these polynomials is equal. However, Shamir's SS is not robust: the cheating parties may release fake shares when recovering the secret. To solve this issue, either of the following verifiable secret sharing (VSS) techniques can be used.

Feldman's VSS [11]**.** Let p be a large prime and g is a generator of a subgroup of $\mathbb{Z}_p^*$ in which the discrete logarithm cannot be solved in polynomial time. To share the secret $s \in \mathbb{Z}_p$, the dealer first generates a $t-1$ degree polynomial $f(x) = a_0 + a_1 x + \cdots + a_{t-1}x^{t-1}$ over $\mathbb{Z}_p$ such that $a_0 = s$. Then, the dealer computes $A_i = g^{a_i}$ for $i \in \{0, 1, \ldots, t-1\}$, and makes these values public. Finally, the dealer computs and sends the share $s_i = f(x_i)$ to each party. Once receiving the share, the party can verify its validity by

$$g^{s_i} = \prod_{j=0}^{t-1} A_j{}^{x_i^j}$$

When recovering the secret, anyone can also use the above equation to verify that the parties have revealed the correct shares.

Pedersen's VSS [24]**.** Let p, q be two large primes such that $q|p-1$, and G is a subgroup of $\mathbb{Z}_p^*$ with order q. Both g and h are elements of G, but nobody knows the value $\log_g h$[1]. To share the secret $s \in \mathbb{Z}_q$, the dealer generates two random polynomials $f(x)$ and $f'(x)$ over $\mathbb{Z}_q$ with degree $t-1$:

$$f(x) = a_0 + a_1 x + \cdots + a_{t-1}x^{t-1} \quad f'(x) = b_0 + b_1 x + \cdots + b_{t-1}x^{t-1}$$

[1] It is crucial that nobody knows the value $\log_g h$. To generate g and h, we first select g in the group G. Then, a distributed coin flipping protocol [3] can be used to generate a random value $r \in \mathbb{Z}_p^*$. Finally, h can be computed as $h = r^{(p-1)/q}$. In case if $h = 1$, we can go back to select another random value $r \in \mathbb{Z}_p^*$ until $h \neq 1$.

where $a_0 = s$. Then the dealer publishes $C_i = g^{a_i} h^{b_i}$ for $i \in \{0, 1, \ldots, t-1\}$. Finally, the dealer computes and sends the share $s_i = f(x_i)$ and $s'_i = f'(x_i)$ to each party. Once receiving the share, the party can verify its validity by

$$g^{s_i} h^{s'_i} = \prod_{j=0}^{t-1} {C_j}^{x_i^j}$$

When recovering the secret, anyone can also use the above equation to verify that the parties have revealed the correct shares.

3 Analysis of Herzberg's PSS Scheme

In this section, we first briefly review Herzberg's PSS scheme [20]. We then show that the secrecy property of Herzberg's scheme might be violated by the mobile adversary in our threat model. We also discuss the impact of this vulnerability with respect to some other PSS schemes.

3.1 Review of Herzberg's PSS Scheme

Denote p as some large prime, and $\{x_1, x_2, \ldots, x_n\}$ be the public index values associated with each party, respectively. In the k-th time period, the secret $s \in \mathbb{Z}_p$ is shared among the parties through the $t-1$ degree polynomial $f^{(k)}(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}$ over $\mathbb{Z}_p$ such that $a_0 = s$. The party P_i holds the share $s_i^{(k)} = f^{(k)}(x_i)$. At the beginning of the $(k+1)$-th time period, the parties will jointly update these shares. And the update phase includes a share recovery protocol followed by a share refreshment protocol.

Share Recovery. The set of parties in Λ, where $|\Lambda| \geq t$, jointly recover the lost share $s_r^{(k)}$ for the party P_r as follows:

1. P_i picks a random $t-1$ degree polynomial $\delta_i(x) = \delta_{i,0} + \delta_{i,1} x + \cdots + \delta_{i,t-1} x^{t-1}$ over $\mathbb{Z}_p$ such that $\delta_i(x_r) = 0$. For example, P_i can first randomly pick the coefficients $\{\delta_{i,j}\}_{j \in \{1,\ldots,t-1\}}$ from $\mathbb{Z}_p$, and then computes $\delta_{i,0} = -\sum_{j=1}^{t-1} \delta_{i,j} {x_r}^j$.
2. P_i computes $u_{i,j} = \delta_i(x_j)$ and sends it to each other party P_j using the secure channel.
3. P_i computes $s'_i = s_i^{(k)} + \sum_{j \in \Lambda} u_{j,i}$ and sends this value to P_r using the secure channel.
4. Finally, P_r uses the received values to interpolate a polynomial $g(x) = f^{(k)}(x) + \sum_{i \in \Lambda} \delta_i(x)$, obtaining $s_r^{(k)} = g(x_r)$.

Because each of the polynomial $\delta_i(x)$ is randomly chosen, P_r cannot learn the polynomial $f^{(k)}(x)$. Hence, P_r cannot learn the secret. And the share $s_r^{(k)}$ is recovered for P_r without being disclosed to the other parties.

Share Refreshment. Each party P_i, $i \in \{1, 2, \ldots, n\}$, performs the share refreshment protocol as follows:

1. P_i picks random values $\{\lambda_{i,j}\}_{j \in \{1,2,\ldots,t-1\}}$ from $\mathbb{Z}_p$. These values define the polynomial $\lambda_i(x) = \lambda_{i,1}x + \cdots + \lambda_{i,t-1}x^{t-1}$ over $\mathbb{Z}_p$ such that $\lambda_i(0) = 0$.
2. P_i computes $w_{i,j} = \lambda_i(x_j)$ and sends it to each other party P_j using the secure channel.
3. P_i computes its new share $s_i^{(k+1)} = s_i^{(k)} + \sum_{j=1}^{n} w_{j,i}$, and erases the old share $s_i^{(k)}$ as well as all the intermediate values. Now, the same secret is shared among the parties through the $t-1$ degree polynomial $f^{(k+1)}(x) = f^{(k)}(x) + \sum_{i=1}^{n} \lambda_i(x)$.

To achieve the robustness property, Feldman's VSS is used both in the share recovery and in the share refreshment, ensuring that the parties have generated and shared the polynomial properly.

3.2 Threat Analysis of Herzberg's Scheme in Our Threat Model

The security of Herzberg's PSS scheme have been proved in [21]. However, the proof relies on the adjacent assumption. Now, we show that if this assumption is removed, as in our threat model, the secrecy property of Herzberg's scheme may be violated by the mobile adversary.

To simplify the description, considering the case that the entire lifetime of the secret is divided into two time periods, as shown in Fig. 1. We allow the mobile adversary to corrupt some parties in time period 1 and then move on to corrupt some other parties in time period 2. Without loss of generality, we assume that the parties $\{P_1, P_3, P_4 \ldots, P_t\}$ are corrupted in time period 1, and the parties $\{P_2, P_3, P_4, \ldots, P_t\}$ are corrupted in time period 2. Because the mobile adversary can learn the corrupted parties' private information, for each polynomial $\lambda_i(x)$ in the share refreshment protocol, the mobile adversary knows that $t-1$ points $(x_2, w_{i,2}), (x_3, w_{i,3}), \ldots, (x_t, w_{i,t})$ will pass this polynomial. In addition, the mobile adversary also knows that the point $(0, 0)$ will pass this polynomial. Therefore, these t points allows the mobile adversary to interpolate the polynomial $\lambda_i(x)$. With the knowledge of all these polynomials $\lambda_i(x)$ for $i \in \{1, 2, \ldots, n\}$, the old shares and the new shares are no longer independent. In other words, the mobile adversary knows how a given share in time period 1 has been updated into time period 2. Therefore, the mobile adversary can combine P_1's share in time period 1 with the $t-1$ shares of $P_2, P_3, \ldots, P_t$ in time period 2 to recover the secret[2].

Note that in the share recovery protocol, the mobile adversary may also find out all polynomials $\delta_i(x)$ for $i \in \Lambda$, because she knows $t-1$ points held by the corrupted parties and an additional point $(x_r, 0)$. Hence, these t points can be used to interpolate each of these polynomials. However, since these polynomials

[2] Note that a similar problem has been independently discovered by Nikov and Nikova in [22]. But its consequences were not elaborated and no solution of this problem was proposed in that work.

are only used privately by the party P_r, the knowledge of these polynomials does not affect the secrecy property in Herzberg's scheme.

3.3 Some Other PSS Schemes in Our Threat Model

In the literature, several other proactive secret sharing schemes are suffering similar vulnerabilities. A common feature of these schemes is that they all use $t-1$ degree polynomials to update the shares. For example, in [21], Jarecki has introduced a scheme that replaces Feldman's VSS with Pedersen's VSS, while the other technical details remain the same as in Herzberg's scheme. In [29], Stinson et al. have introduced an unconditional secure proactive secret sharing scheme, in which a $t-1$ degree symmetric bivariate function is used to refresh the shares. In [27], Schultz et al. have introduced a PSS scheme that allows on-the-fly reconfiguration of the threshold. In Schultz's scheme, the share recovery protocol is combined with the share refreshment protocol, and $t-1$ degree polynomials are used to refresh the shares. Therefore, the secrecy property in these scheme also might be violated by the mobile adversary in our threat model.

The above discussions may give the readers a false intuition that any PSS scheme using $t-1$ degree polynomials to update the shares is vulnerable in our threat model. A counterexample is that although Ostrovsky and Yung [23] also have used $t-1$ degree polynomials to update the shares, the above threat analysis does not apply to it. Because Ostrovsky's scheme has used two layers of SS, while the other vulnerable schemes only use one layer of SS. For similar reasons, this threat analysis is not directly applicable with Frankel's scheme [12] or Rabin's scheme [25]. However, we note that these schemes are not specially designed to withstand the mentioned attack, and it is still unknown whether their security can be formally proved in our new threat model.

4 Modification of Herzberg's PSS Scheme

In this section, we modify Herzberg's PSS scheme, making it secure against the mobile adversary in our threat model. Because the share recovery protocol in Herzberg's scheme does not suffer the vulnerability discussed in the previous section, we focus our description on the share refreshment protocol.

As a high level overview, we use $2t-1$ degree random polynomials with 0 in the constant coefficient to refresh the shares. Hence, the mobile adversary who corrupts $t-1$ parties cannot learn any information of these polynomials. However, after adding these $2t-1$ degree random polynomials with the original $t-1$ degree polynomial that shares the secret, the result polynomial will have a degree $2t-1$ rather than $t-1$. But this implies that the secret cannot be recovered by any t parties, violating the optimal resilience property. To further address this issue, the parties need to jointly truncate the $2t-1$ degree polynomial into a $t-1$ degree polynomial with the constant coefficient unchanged.

4.1 Jointly Polynomial Truncation

In [3], Ben-Or et al. have introduced a novel technique to jointly truncate polynomials. We adapt this method in our proposed scheme with two necessary changes. Firstly, in Ben-Or's scheme, a $2t$ degree polynomial is truncated into a t degree polynomial with the first $t+1$ coefficients unchanged. While in our proposed scheme, a $2t-1$ degree polynomial is truncated into a $t-1$ degree polynomial with only the constant coefficient unchanged. In other words, we truncate the polynomial in a randomised fashion instead of a fixed one. And we show later that this change is crucial for the security of our proposed scheme. Secondly, to ensure the robustness property, error correction codes are used in Ben-Or's scheme, but we use VSS instead in order to achieve the optimal resilience property.

Lemma 1. *For $i \in \{1, 2, \ldots, n\}$, suppose a_i is some public constant and z_i is the private input of the party P_i, then the linear function $F(z_1, z_2, \ldots, z_n) = a_1 z_1 + a_2 z_2 + \cdots + a_n z_n$ can be computed by the parties in a secure and distributed fashion.*

Proof. (Sketch) Firstly, each party P_i shares its private input z_i among the parties using (t, n)-threshold verifiable secret sharing. Denote $s_{i,j}$ as the share of z_i held by the party P_j. Then $a_1 s_{1,j} + a_2 s_{2,j} + \cdots + a_n s_{n,j}$ will be the corresponding share of $a_1 z_1 + a_2 z_2 + \cdots + a_n z_n$ held by P_j, thanks to the homomorphic property of secret sharing [4]. If the result is supposed to be made public, each party broadcasts its computed share and anyone can retrieve the result by polynomial interpolation. And if the result is supposed to be known by some certain party, then each party sends its computed share to this party using the secure channel. Therefore, the function $F(z_1, z_2, \ldots, z_n)$ can be computed in a secure and distributed fashion. Here, the word "secure" implies both correctness and secrecy. Correctness means that if the private inputs are properly shared, the correct result can always be computed even in the presence of any minority of cheating parties, and this property can be ensured using VSS. Secrecy means that apart from the final result, the adversary who corrupts any minority of parties learns no additional information.

Lemma 2. *Suppose M is a public $n \times n$ matrix. For $i \in \{1, 2, \ldots, n\}$, z_i is the private input of the party P_i. Denote Z as a vector $[z_1, z_2, \ldots, z_n]$ and Y as another vector $[y_1, y_2, \ldots, y_n]$. Then $Y = Z \cdot \mathsf{M}$ can be computed in a secure and distributed fashion, such that by the end of the computation, each party P_i obtains the value y_i without leaking any other information.*

Proof. Since y_i is the vector Z times the i-th column of the matrix M. It can be computed in a secure and distributed fashion by Lemma 1. By the end of the computation, each party sends its computed share to the party P_i using the secure channels. Hence, only P_i knows the value y_i. If this process is repeated for $i \in \{1, 2, \ldots, n\}$, the desired computation can be carried out in a secure and distributed fashion.

Theorem 1. *Suppose* $h(x) = h_0 + h_1 x + h_2 x^2 + \cdots + h_{2t-1} x^{2t-1}$ *is a polynomial with degree* $2t-1$*, and each party* P_i *holds a share of* $h(x)$ *as* $s_i = h(x_i)$*. Then, these parties can jointly truncate* $h(x)$ *into a* $t-1$ *degree polynomial* $k(x) = k_0 + k_1 x + \cdots + k_{t-1} x^{t-1}$ *in a secure and distributed fashion with the constant coefficient unchanged, i.e.* $h_0 = k_0$*. By the end of the computation, each party holds a share of* $k(x)$ *as* $r_i = k(x_i)$*.*

Proof. (Sketch) Denote B as an $n \times n$ Vandermonde matrix

$$\mathsf{B} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \dots & x_n^{n-1} \end{pmatrix}$$

$H = [h_0, h_1, \dots h_{2t-1}, \dots, 0]$ is an n-vector that represents the coefficients of $h(x)$; $K = [k_0, k_1, \dots k_{t-1}, 0, \dots, 0]$ is an n-vector that represents the coefficient of $k(x)$; the $n \times n$ projection matrix P satisfies $H \cdot \mathsf{P} = K$ (i.e. the first column is a vector led by 1 and followed by 0s, the second column till the t-th column are random vectors generated in a distributed fashion and shared among the parties [16], and the other columns are all zero vectors); $S = [s_1, s_2, \dots, s_n]$ is an n-vector that represents the shares of $h(x)$; $R = [r_1, r_2, \dots, r_n]$ is an n-vector that represents the shares of $k(x)$. Hence, we have $H \cdot \mathsf{B} = S$ and $K \cdot \mathsf{B} = R$. Moreover, because B is a Vandermonde matrix, it is always reversible as its determinant cannot be 0. Therefore, we have $S \cdot (\mathsf{B}^{-1} \cdot \mathsf{P} \cdot \mathsf{B}) = R$. Denote $\mathsf{T} = \mathsf{B}^{-1} \cdot \mathsf{P} \cdot \mathsf{B}$, we have $S \cdot \mathsf{T} = R$, where T can be jointly computed in a secure and distributed fashion. By Lemma 2, the $2t-1$ degree polynomial $h(x)$ can be truncated into a $t-1$ degree $k(x)$ in a secure and distributed fashion with the constant coefficient unchanged.

4.2 Our Proposed Scheme

Our proposed scheme works as follows: denote p as a large prime and g is a generator of a subgroup of $\mathbb{Z}_p^*$ in which the discrete logarithm cannot be solved in polynomial time. Suppose in the k-th time period, the secret $s \in \mathbb{Z}_p$ is shared among the parties $P_1, P_2, \dots, P_n$ using a $t-1$ degree polynomial $f^{(k)}(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}$ over $\mathbb{Z}_p$ such that $s = a_0$, and the commitments $A_i = g^{a_i}$ for $i \in \{0, 1, \dots, t-1\}$ are made public. Each party P_i's secret share is $s_i^{(k)} = f^{(k)}(x_i)$, and P_i can verify the validity of its share by

$$g^{s_i^{(k)}} = \prod_{j=0}^{t-1} {A_j}^{x_i^j}$$

In the share refreshment, each party P_i, for $i \in \{1, 2, \dots, n\}$, performs as follows:

1. P_i generates a random $2t-1$ degree polynomial as $\lambda_i(x) = \lambda_{i,1} x + \lambda_{i,2} x^2 + \cdots + \lambda_{i,2t-1} x^{2t-1}$ over $\mathbb{Z}_p$ such that $\lambda_i(0) = 0$. P_i also broadcasts $B_{i,j} = g^{\lambda_{i,j}}$ for $j \in \{1, 2, \dots, 2t-1\}$.

2. P_i computes $w_{i,j} = \lambda_i(x_j)$ and sends it to each other parties P_j using the secure channel. P_i can verify whether its received share $w_{j,i}$ from each other party is valid by
$$g^{w_{j,i}} = \prod_{k=1}^{2t-1} B_{j,k}{}^{x_i^k}$$
3. In order to achieve the optimal resilience property, once receiving the value $w_{j,i}$, P_i needs to further share this value among the parties in a (t, n)-threshold fashion. If some parties are found faulty in Step 2 or in Step 3, they will be disqualified from the protocol and their polynomials will be excluded. At this moment, the set of the remaining parties is denoted as Λ.
4. P_i computes $s_i = s_i^{(k)} + \sum_{j\in\Lambda} w_{j,i}$, and this value is a share of the $2t-1$ degree polynomial
$$h(x) = f^{(k)}(x) + \sum_{j\in\Lambda} \lambda_j(x) = h_0 + h_1 x + \cdots + h_{2t-1}x^{2t-1}$$
The commitments of $h(x)$ can be publicly computed as $C_i = g^{h_i} = A_i \cdot \prod_{j\in\Lambda} B_{j,i}$ for $i \in \{0, 1, \ldots, t-1\}$ and $C_I = g^{h_I} = \prod_{j\in\Lambda} B_{j,I}$ for $I \in \{t, t+1, \ldots, 2t-1\}$.
5. Finally, the parties jointly truncate the $2t-1$ degree polynomial $h(x)$ into a $t-1$ degree polynomial $k(x)$ with the constant coefficient unchanged. Denote $S = [s_1, s_2, \ldots, s_n]$ as the n-vector that represents the shares of $h(x)$, and $R = [r_1, r_2, \ldots, r_n]$ as the n-vector that represents the shares of $k(x)$, the truncation is done by $S \cdot \mathsf{T} = R$, where T can be computed in a secure and distributed fashion as shown in Theorem 1. Now, $k(x)$ is the updated polynomial that will be used in the $(k+1)$-th time period as $f^{(k+1)}(x)$, and each party P_i holds a share $s_i^{(k+1)} = r_i$. Note that if any party P_i is found cheating in this step, the corresponding share s_i will be recovered by the uncorrupted parties. And this ensures that this step will always finish successfully.

4.3 Security Analysis

Theorem 2. *Our proposed PSS scheme is robust and secret in the presence of the mobile adversary who has the ability to corrupt any minority of the parties.*

Proof. We prove this theorem using the inductive method. Firstly, we assume that at initialisation of the protocol, the secret is properly shared among the parties through (t, n)-threshold secret sharing. Furthermore, we assume that in each time period $1, 2, \ldots, k$, the above theorem holds. And we prove that in the time period $k+1$, the adversary who has the ability to corrupt any minority of the parties can neither prevent the secret from being recovered nor learn any information of the secret.

Robustness: In each time period, the validity of the shares can be verified using the public commitments. Without loss of generality, we assume that the

shares $s_1^{(k+1)}, s_2^{(k+1)}, \ldots, s_t^{(k+1)}$ are valid and they will be used to recover the secret. Denote $L_i = \prod_{j=1, j\neq i}^{t} x_j/(x_j - x_i)$ as the Lagrange coefficients for $i \in \{1, 2, \ldots, t\}$. Then we have

$$\sum_{i=1}^{t} s_i^{(k+1)} \cdot L_i = f^{(k+1)}(0) = f^{(k)}(0) + \sum_{j\in\Lambda} \lambda'_j(0) = s$$

where $\lambda'_j(x) = \lambda'_{i,1}x + \lambda'_{i,2}x^2 + \cdots + \lambda'_{i,t-1}x^{t-1}$. Although the polynomials $\lambda'_j(x)$ and $\lambda_j(x)$ are independent because of the randomised truncation, the equation $\lambda'_j(0) = \lambda_j(0) = 0$ always holds for all $j \in \Lambda$. Therefore, based on the assumption that the mobile adversary cannot corrupt more than $t-1$ parties in time period $k+1$, there exists at least t uncorrupted parties and the secret can be correctly recovered.

Secrecy: To prove that the proposed scheme achieves the secrecy property. We prove that there exists a PPT simulator $\mathcal{SIM}$ who can simulate the mobile adversary's view in share refreshment, and the simulated view is indistinguishable from the one in the real run of the protocol. Without loss of generality, we assume that the parties $P_1, P_2, \ldots, P_{t-1}$ are corrupted and the mobile adversary knows their shares $s_1^{(k)}, s_2^{(k)}, \ldots, s_{t-1}^{(k)}$. The simulator $\mathcal{SIM}$ works as follows:

1. Each party P_i generates a random $2t-1$ degree polynomial as $\widetilde{\lambda_i}(x) = \widetilde{\lambda_{i,1}}x + \widetilde{\lambda_{i,2}}x^2 + \cdots + \widetilde{\lambda_{i,2t-1}}x^{2t-1}$ over $\mathbb{Z}_p$ such that $\widetilde{\lambda_i}(0) = 0$. P_i also broadcasts $\widetilde{B_{i,j}} = g^{\widetilde{\lambda_{i,j}}}$ for $j \in \{1, 2, \ldots, 2t-1\}$.
2. P_i computes $\widetilde{w_{i,j}} = \widetilde{\lambda_i}(x_j)$ and sends it to each other party P_j using the secure channel. P_i can verify the validity of $\widetilde{w_{j,i}}$ using the public commitments $\widetilde{B_{j,k}}$ for $k \in \{1, 2, \ldots, 2t-1\}$. Those values received by the corrupted parties are forwarded to the mobile adversary.
3. Once receiving the value $\widetilde{w_{j,i}}$, P_i further shares this value among the parties using (t, n)-threshold verifiable secret sharing. Similarly, any cheating party will be disqualified, and the set of the remaining parties is denoted as Λ.
4. In this step, the simulator $\mathcal{SIM}$ computes $s_i = s_i^{(k)} + \sum_{j\in\Lambda} \widetilde{w_{j,i}}$ for $i \in \{1, 2, \ldots, t-1\}$, and it sends these values to the mobile adversary.
5. In order to truncate the $2t-1$ degree polynomial $h(x)$ into a $t-1$ degree polynomial $k(x)$ with the constant coefficient unchanged, each party P_i needs to share its value s_i among the parties in a (t, n)-threshold fashion. For those corrupted parties, the simulator $\mathcal{SIM}$ can share their values in the normal way. However, $\mathcal{SIM}$ does not know the values $s_t, s_{t+1}, \ldots, s_n$. To simulate the (t, n)-threshold secret sharing of these values, the simulator $\mathcal{SIM}$ computes

$$g^{s_i} = g^{s_i^{(k)}} \cdot \prod_{l\in\Lambda} g^{\widetilde{w_{l,i}}} = \prod_{j=0}^{t-1} {A_j}^{x_i^j} \cdot \prod_{l\in\Lambda} g^{\widetilde{w_{l,i}}}$$

for $i \in \{t, t+1, \ldots, n\}$. Then, for each of the value $\{g^{s_i}\}_{i\in\{t,t+1,\ldots,n\}}$, $\mathcal{SIM}$ selects $t-1$ random values $\{\epsilon_i\}_{i\in\{1,2,\ldots,t-1\}}$ and sends these values to the $t-1$ corrupted parties respectively. Denote $\mathcal{M}$ as the following $t \times t$ matrix

$$\mathsf{M} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & x_1 & x_1^2 & \dots & x_1^{t-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{t-1} \\ & & & \vdots & \\ 1 & x_{t-1} & x_{t-1}^2 & \dots & x_{t-1}^{t-1} \end{pmatrix}$$

and $\sigma_{i,j}$ is the (i,j)-th entry of M^{-1}. Now, $\mathcal{SIM}$ broadcasts the commitments $\{D_i\}_{i\in\{0,1,\dots,t-1\}}$, where $D_0 = g^{s_i}$ and $D_j = (g^{s_i})^{\sigma_{j,1}} \cdot \prod_{l=1}^{t-1}(g^{\epsilon_l})^{\sigma_{j,l+1}}$ for $j \in \{1,2,\dots,t-1\}$. Note that these commitments ensures that the mobile adversary will accept the values $\{\epsilon_i\}_{i\in\{1,2,\dots,t-1\}}$ as the shares of s_i. As follows, the parties jointly trancate the polynomial $h(x)$ into $k(x)$.

It is obvious that the above simulation can finish in polynomial time. Now, we show that the mobile adversary cannot distinguish the above simulated conversation from a real run of the protocol.

- **Indistinguishability of information in Step 1 and 2:** the $2t-1$ degree polynomials $\{\lambda_i(x)\}$ and $\{\widetilde{\lambda_i}(x)\}$ for $i \in \{1,2,\dots,n\}$ are randomly selected both in the real protocol and in the simulation. Hence, they are indistinguishable.
- **Indistinguishability of information in Step 3:** both the real protocol and the simulation share the values $w_{j,i}$ and $\widetilde{w_{j,i}}$ among the parties using a random $t-1$ degree polynomial. Hence, they are indistinguishable.
- **Indistinguishability of information in Step 4:** the s_i values hold by the corrupted parties are randomly distributed in $\mathbb{Z}_p$ both in the real protocol and in the simulation. Hence, they are indistinguishable.
- **Indistinguishability of information in Step 5:** the mobile adversary's view of sharing the values $\{s_i\}_{i\in\{1,2,\dots,n\}}$ is consistent both in the real protocol and in the simulation. Moreover, by Theorem 1, the joint polynomial truncation can be done in a secure and distributed fashion. Hence, the real protocol and the simulation in this step is also indistinguishable.

Therefore, the simulated view cannot be distinguished from the one in the real run of the protocol. In other words, the mobile adversary cannot learn any information of the secret in time period $k+1$.

4.4 Some Discussions

Once the reason is clear why Herzberg's PSS scheme fails to maintain its security in our new threat model, it is quite natural to come up with the idea of using polynomials with higher degrees to update the shares in the share refreshment and then truncating the resulting polynomial to the desirable degree. However, we show that if one uses Ben-Or's polynomial truncation method directly, the construction still suffers the same problem as in Herzberg's PSS scheme.

Ben-Or's original method is also capable of truncating a $2t-1$ degree polynomial into a $t-1$ degree polynomial. But it keeps the first t coefficients unchanged rather than just keeping the constant coefficient unchanged as in our proposed scheme. Recall that $\lambda(x)$ is the polynomial used to refresh the shares, $h(x)$ is the polynomial before truncation and $k(x)$ is the polynomial after the truncation. Their shares are w_i, s_i, r_i, respectively. Denote $p(x)$ as a polynomial with degree $t-1$ containing the first t coefficients of the polynomial $\lambda(x)$. The relationship $h(x)-k(x) = \lambda(x)-p(x)$ always holds if Ben-Or's polynomial truncation method is used directly. Thanks to the homomorphic property of SS, the value $w_i+r_i-s_i$ represents a share for the polynomial $p(x)$. Therefore, if the mobile adversary $\mathcal{A}$ is assumed to corrupt $t-1$ parties, $\mathcal{A}$ can obtain $t-1$ shares of $p(x)$. And this implies that $\mathcal{A}$ is able to launch the same attack as shown in Sect. 3.2. This is why we have adapted a variant of Ben-Or's method in our proposed scheme so that the truncation is performed in the randomised fashion instead of a fixed one.

5 Conclusion

In this paper, we revisited the research of provably secure and optimal resilient PSS. We discussed the negative aspects caused by the adjacent assumption which is widely used in the existing PSS schemes. And this motivates us to consider whether this assumption can be removed from the threat model in PSS schemes. However, we showed that if it is removed, many existing schemes will become insecure. We then used the polynomial truncation method to improve Herzberg's PSS scheme, making it secure without the adjacent assumption. To the best of our knowledge, this is the first PSS scheme satisfying this feature.

Acknowledgement. This work was partially supported by the National Natural Science Foundation of China (Grant No. 61572303, 61772326, 61822202, 61672010, 61702168, 61872087). We are very grateful to the anonymous reviewers for pointing out an error in a previous version of this paper as well as many valuable comments.

References

1. Almansa, J.F., Damgård, I., Nielsen, J.B.: Simplified threshold RSA with adaptive and proactive security. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 593–611. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_35
2. Baron, J., Defrawy, K., Lampkins, J., Ostrovsky, R.: How to withstand mobile virus attacks, revisited. In: ACM Symposium on Principles of Distributed Computing (PODC 2014), pp. 293–302 (2014)
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the 20th ACM Symposium on Theory of Computing (STOC 1988), pp. 1–10 (1988)
4. Benaloh, J.C.: Secret sharing homomorphisms: keeping shares of a secret secret (extended abstract). In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 251–260. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_19

5. Blakley, R.: Safeguarding cryptographic keys. In: Proceedings of the National Computer Conference, vol. 48, pp. 313–317 (1979)
6. Cachin, C., Kursawe, K., Lysyanskaya, A., Strobl, R.: Asynchronous verifiable secret sharing and proactive cryptosystems. In: 9th ACM Conference on Computer and Communication Security (CCS 2002), pp. 88–97 (2002)
7. Canetti, R., Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Adaptive security for threshold cryptosystems. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 98–116. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_7
8. Canetti, R., Halevi, S., Herzberg, A.: Maintaining authenticated communication in the presence of break-ins. In: Proceedings of the 16th ACM Symposium on Principles of Distributed Computing (PODC 1997), pp. 15–24 (1997)
9. Canetti, R., Herzberg, A.: Maintaining security in the presence of transient faults. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 425–438. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_38
10. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: Proceedings of the 20th ACM Symposium on Theory of Computing (STOC 1988), pp. 11–19 (1988)
11. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: Proceedings of the 28th IEEE Symposium on Foundation of Computer Science (FOCS 1987), pp. 427–437 (1987)
12. Frankel, Y., Gemmell, P., MacKenzie, P., Yung, M.: Optimal-resilience proactive public-key cryptosystems. In: Proceedings of the 38th IEEE Symposium on the Foundations of Computer Science (FOCS 1997), pp. 384–393 (1997)
13. Frankel, Y., Gemmell, P., MacKenzie, P.D., Yung, M.: Proactive RSA. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 440–454. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052254
14. Frankel, Y., MacKenzie, P., Yung, M.: Adaptively-secure optimal-resilience proactive RSA. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 180–194. Springer, Heidelberg (1999). https://doi.org/10.1007/978-3-540-48000-6_15
15. Frankel, Y., MacKenzie, P.D., Yung, M.: Adaptive security for the additive-sharing based proactive RSA. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 240–263. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44586-2_18
16. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. J. Cryptol. **1**, 51–83 (2007)
17. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game, or a completeness theorem for protocols with honest majority. In: Proceedings of the 19th ACM Symposium on Theory of Computing (STOC 1987), pp. 218–229 (1987)
18. Hegland, A., Winjum, E., Mjolsnes, S., Rong, C., Kure, O., Spilling, P.: A survey of key management in ad hoc networks. IEEE Commun. **8**(3), 48–66 (2006)
19. Herzberg, A., Jakobsson, M., Jarecki, S., Krawczyk, H., Yung, M.: Proactive public key and signature systems. In: 4th ACM Conference on Computer and Communication Security (CCS 1997), pp. 100–110 (1997)
20. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: how to cope with perpetual leakage. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 339–352. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-44750-4_27
21. Jarecki, S.: Proactive secret sharing and public key cryptosystems. Master's thesis, Department of Electrical Engineering and Computer Science, MIT (1995)

22. Nikov, V., Nikova, S.: On proactive secret sharing schemes. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 308–325. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30564-4_22
23. Ostrovsky, R., Yung, M.: How to withstand mobile virus attacks. In: Proceedings of the 10th ACM Symposium on the Principle of Distributed Computing (PODC 1991), pp. 51–61 (1991)
24. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_9
25. Rabin, T.: A simplified approach to threshold and proactive RSA. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 89–104. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055722
26. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: Proceedings of the 21st ACM Symposium on Theory of Computing (STOC 1989), pp. 73–85 (1989)
27. Schultz, D., Liskov, B., Liskov, M.: MPSS: mobile proactive secret sharing. ACM Trans. Inf. Syst. Secur. **13**(4), 34 (2010)
28. Shamir, A.: How to share a secret. In: Proceedings of 22nd Communication of ACM, pp. 612–613 (1979)
29. Stinson, D.R., Wei, R.: Unconditionally secure proactive secret sharing scheme with combinatorial structures. In: Heys, H., Adams, C. (eds.) SAC 1999. LNCS, vol. 1758, pp. 200–214. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-46513-8_15
30. Yung, M.: The "mobile adversary" paradigm in distributed computation and systems. In: ACM Symposium on Principles of Distributed Computing (PODC 2015), pp. 171–172 (2015)
31. Zhou, L., Haas, Z.: Securing ad hoc networks. IEEE Netw. **13**, 24–30 (1999)
32. Zhou, L., Schneider, F., Renesse, R.: APSS: proactive secret sharing in asynchronous systems. ACM Trans. Inf. Syst. Secur. **8**(3), 259–286 (2005)