A Novel Adaptive Proactive Secret Sharing without a Trusted Party

Xiuqun Wang

Beijing Certificate Authority Ltd., Beijing, 100080, P. R. China
xqwang91@hotmail.com

Abstract. A $(t+1,n)$ proactive secret sharing is to protect a secret in long-lived system by distributing it to a group of n participants and refreshing their shares periodically in this fixed group, while any $t+1$ and more than $t+1$ shares can reconstruct the secret. In some environment, it needs to change not only the number of participants n but also the threshold value t. An adaptive proactive secret sharing is to refresh the shares as t and n change. In this paper, we propose a novel adaptive proactive secret sharing scheme without a trusted party. Our proposed scheme is uniformly efficient and tolerates t Byzantine faults in any single time interval, where the number of participants $n \geq 3t+1$. The threshold value t and the number of participants n can be changed arbitrarily in two adjacent intervals. We also prove that our proposed scheme is secure under the discrete logarithm intractability assumption.

Keywords. Secret sharing, proactive secret sharing, adaptive proactive secret sharing, validated Byzantine agreement protocol.

1 Introduction

Secret sharing [22, 2] is a basic cryptographic protocol, which can protect a secret by distributing it among different participants. In a $(t+1,n)$ threshold secret sharing, the secret s is distributed among n participants, each holding a share. Every group of $t+1$ participants can recover the secret s, while any smaller group of participants cannot get any information about the secret s. In order to reconstruct the secret, the adversary has to corrupt at least $t+1$ participants. Comparing the threshold secret sharing with the secret owned by a single participant, the security of secret sharing is significantly enhanced.

However, for the long-lived secrets, the adversary may still have enough time to gradually corrupt enough participants. A natural defense is to periodically update the secret, but it is not always possible, e.g. for the long-lived certification authority (CA) signing key. To address this problem, Herzberg et al. [13] introduced the concept of proactive secret sharing (PSS). In a proactive secret sharing, the lifetime of the secret is divided into time intervals. The shares are periodically refreshed at the beginning of each time interval using the so-called update algorithm, after which new shares of the same secret are obtained by the participants, the old shares are discarded safely. Their proposed PSS is secure against a mobile adversary [17] who can corrupt each participant many times but it cannot obtain the secret unless it has been able to corrupt at least $t+1$ participants in a single time interval.

The traditional PSS assumes that the threshold value t is the same in any interval. However, if the security environment changes, then we need adjust the threshold value t accordingly. For example, a discovery of new vulnerability in the operating system might lead to an increase of t, whereas the removal of the the vulnerability might lead to a decrease of t. Adaptive proactive secret sharing [21, 24] (also called the mobile proactive secret sharing) allows to increase or decrease the threshold value t and adjust the number n of participants.

1.1 Related Work

The first threshold secret sharing schemes were proposed by Shamir [22] and Blakey [2]. In order to prevent an inconsistent secret sharing, Feldman [8] and Pedersen [20] introduced the verifiable secret sharing (VSS) scheme based on Shamir secret sharing. Their schemes need a trusted party to distribute the secret. Pedersen [19] showed that it is possible to design threshold cryptosystems, in which the role of the trusted party is distributed among all parties.

The concept of proactivity was introduced by Ostrovsky and Yung [17] in the *mobile adversary* context. The solution discussed in their paper achieved the information theoretic security in the presence of mobile adversary attacks. Canetti and Herzberg [6] gave a practical solution for proactive distributed pseudorandom generator. Herzberg et al. [13] applied the concept of proactivity to secret sharing. This work was continued by Alon et al. in [1].

The proactive secret sharing have been studied extensively in the literature [13, 25, 3, 21, 24, 23]. The PSS schemes of Herzberg et al. [13] and Stinson et al. [23] need a distributed commitment protocol in the update phase, in which the participants are committed to the value of “0” in order to update the participant shares. Nikov et al. [16] showed that these schemes are vulnerable to an attack. Frankel et al. [10] used the re-sharing protocol in order to refresh the secret. Zhou et al. [25] used (l, l) secret sharing [14], where $l = \binom{n}{t}$, to build an asynchronous secret sharing, which can tolerate Byzantine faults, with the communication complexity $O(\binom{n}{t})$. Cachin et al. [3] considered asynchronous secret sharing using bivariate polynomials, which can tolerate Byzantine faults as well, with communication complexity $O(n^3)$. In the update phase, the Cachin et al. scheme uses a Byzantine agreement to determine the participants contributions to the new shares.

Recently, Wang et al. [24] and Schultz et al. [21] considered the adaptive proactive secret sharing (APSS), in these schemes they not only change the number n of participants but also change the threshold t. These two APSS schemes use the idea of Herzberg et al. PSS [13] to update the shares by committing to the value 0, which was proven to be insecure in the mobile attack model.

All these PSS and APSS schemes initiate the secret using a trusted party. This is certainly a security problem if the trusted party becomes corrupted. In this paper, we propose an adaptive proactive secret sharing without a trusted party in the asynchronous communication system.

1.2 Contribution

Our contribution is the design of a novel adaptive proactive secret sharing without a trusted party in asynchronous communication system. Our scheme is similar with Cachin et al. scheme [3], but it initiates the secret without a trusted party. In the initialization phase of our proposed scheme, every participant P_i chooses a random value s_i and distributes it among all participants, the secret x is a function as $x = \sum_{i \in L} s_i \lambda_L^i$, where L is the set of participants agreed by a Byzantine agreement, λ_L^i is the Lagrange coefficient according to L (see Sec. 3.1). In other schemes [19], the secret is generated by the function as $x = \sum_{i \in B} s_i$ where B is some subset of all the participants. Our function can guarantee the robustness of our proposed adaptive proactive secret sharing which doesn't require a trusted party in the initialization phase (see Sec. 4.1).

Then, each participant updates their previous share using re-sharing technique, and generates the new share using the same function (see Sec. 3.2). With this method, we can change the threshold value and the number of participants arbitrarily in each interval (see Sec. 3.3). Our method is simpler than that in the schemes [24, 21] which have to distinguish the change of the threshold value into two cases (increase and decrease).

Our proposed scheme is efficient and tolerates Byzantine faults as long as $n \geq 3t + 1$. The expected message complexity is $O(n^3)$ and communication complexity is $O((n)^3 t + (n)^2 t^3)$. We also prove that our proposed scheme is secure under the discrete logarithm intractability assumption (see Sec. 4).

1.3 Organization

The rest of the paper is organized as follows. In Section 2, we introduce the basic security model and definitions. Section 3 presents our adaptive proactive secret sharing scheme without a trusted party. Section 4 gives the proof of the security and discusses the complexity of the scheme. We conclude in Section 5.

2 Model and Definitions

2.1 Security Model

We use the security model given in [12]. The adversary can not only try to know secret information (passive adversary also called honest but curious) but also can generate Byzantine faults (malicious adversary that deviates arbitrarily from the protocol). The model used in our work consists of the following components.

Participants and Communication. We assume that our system consists of a set of n participants, $\{P_1, P_2, \ldots, P_n\}$, which can be modeled by a polynomial-time Turing machine. The participants are connected by a complete network of secure point-to-point channels, which can be implemented with standard cryptographic techniques.

The participants have access to a multicast channel C, with the property, that messages sent on C reach every participant connected to it. We also assume that the communication channels provide an asynchronous message delivery. This means that messages sent on either a point-to-point or a multicast channel are received by the recipients finally although they will be relayed. A failure of a communication channel to deliver a message can be treated as a failure of the sender.

The life-time of the whole system is divided into time intervals (which are determined by the common global clock) such as a day, a week, etc. Each time interval starts from an update (or change of threshold) phase, during which the participants engage in an interactive update (change of threshold) algorithm and after which each participant holds a new share of the secret.

Adversary. We assume that the adversary can corrupt up to t out of n participants in the network. A Byzantine adversary [15], who also called malicious adversary, can corrupt participants and deviate from the protocol in an arbitrary way.

We assume that the adversary is computationally bounded, i.e. he cannot break the underlying computational assumptions that are used in the construction. There are two kinds of adversaries: static and mobile. A *static adversary* chooses participants that it will try to corrupt at the beginning of the protocol. A *mobile adversary* can corrupt up to t participants during a single time interval. From now on, we consider the mobile Byzantine adversary. Note that we assume that at the beginning of each time interval, the adversary has been removed by the update algorithm.

Given a protocol P, the view of an adversary A, denoted by $View_A(P)$, is defined by the knowledge of the adversary, i.e., the transcript of past run of the protocol that includes the communication and input/output observations, the computational and memory history of all the corrupted participants, the public communications and outputs of the protocol.

2.2 Definitions

Adaptive Proactive Secret Sharing is a collection of four algorithms $(\mathcal{I}, \mathcal{U}, \mathcal{C}, \mathcal{R})$ performed jointly by the participants $\{P_1, \ldots, P_n\}$.

The initialization algorithm $\mathcal{I}$ sets up the scheme and it is run jointly by all participants. At the beginning of the algorithm $\mathcal{I}$, each participant P_i generates a random integer s_i as input and at the end, each participant P_i gets a share x_i of the secret x.

If there is no threshold change, then the participants perform jointly the update algorithm $\mathcal{U}$ at the beginning of the current time interval. Each P_i provides the private share x_i from the previous interval as the input to the algorithm and the participant P_j gets a new private share $\bar{x}_j$ for the current interval. The elements $(\bar{x}_1, \ldots, \bar{x}_n)$ are shares of the same secret x.

When there is a threshold change, the threshold changes from t to t', and the number of participants change from n to n'. At the beginning of the current time interval, the participants perform jointly the threshold change algorithm $\mathcal{C}$. Each P_i provides the private share x_i from the previous interval as input to the algorithm and every P_j gets a new private share $\bar{x}_j$ for the current interval. The elements $(\bar{x}_1, \ldots, \bar{x}_{n'})$ are shares of the same secret x.

The algorithm $\mathcal{R}$ is run jointly by a subset of participants. If the number of participants is at least $t + 1$ and they work together with their correct shares, then $\mathcal{R}$ reconstructs the secret x. Otherwise, $\mathcal{R}$ fails.

Definition 1. *Let $(\mathcal{I},\mathcal{U},\mathcal{C},\mathcal{R})$ be a collection of four algorithms introduced above. Assume further that the threshold changes from t to t' for current interval, and the number of participants changes from n to n'. Given a mobile adversary who corrupts at most t participants at each time interval and has access to the view of the algorithm $\mathcal{I}$ as well as to the views of multiple runs of the algorithms $\mathcal{U}$ and $\mathcal{C}$, then we call it a secure adaptive proactive secret sharing if the following properties hold:*

– ***Robustness:*** *The collection of $t+1$ participants who jointly execute the algorithm $\mathcal{R}$ in any interval can always reconstructs the secret x.*
– ***Privacy:*** *An adversary knows no information about secret x.*
– ***Liveness:*** *Execution of the protocol always terminates on all servers that are honest.*

2.3 Computational Assumptions

Let p and q be two large primes, such that $q \mid p - 1$ and $q > n$, where n is the cardinality of the set of participants. Assume further that G denotes a multiplicative subgroup of order q of $\mathbb{Z}_p$, and g, h are two distinct generators of G such that no one knows $\log_g h$.

We use the well-known discrete logarithm (DL) intractability assumption which can be worded as follows. Given a cyclic group G, its generator g, and a random element $h \in G$. Find an element $\alpha \in \mathbb{Z}_q$ such that $h = g^{\alpha}$, in other words, find $\alpha = \log_g h$. It is believed that there is no polynomial time algorithm that solves DL assumption with a non-negligible probability.

2.4 Multi-valued Validated Byzantine Agreement

Byzantine agreement [15, 18] is a fundamental problem in distributed computing. In asynchronous networks, it is impossible to solve it by deterministic protocols [9], which means that one has to resort to a randomized protocol. A polynomial-time solution to this problem was given by Canetti and Rabin [7]. The standard solution of the Byzantine agreement implements only a binary decision in asynchronous networks. The multi-valued validated Byzantine agreement (VBA) [4] extends this to arbitrary domain by means of the so-called external validity condition. It is based on a global, polynomial-time computable predicate Q_{ID} known to all participants. Each participant may propose a value that perhaps contains validation information. The agreement ensures that the decision value satisfies Q_{ID} and that it has been proposed by at least one participant.

When a participant P_i starts a VBA with a tag ID and input $v \in \{0,1\}^*$, we say that P_i proposes v_i for ID (the honest participants propose values that satisfy Q_{ID}). When a participant P_i terminates a VBA with a tag ID and outputs value v, we say that P_i decides the value v.

Definition 2. *Multi-valued validated Byzantine agreement – a protocol solves VBA with predicate Q_{ID} if it satisfies almost always the following conditions (i.e. the conditions fail with a negligible probability).*

External validity: *Any honest participant who terminates decides v for ID such that $Q_{ID}(v)$ holds.*

Agreement: *If some honest participant decides v for ID, then all honest participants who terminate decide v for ID.*

Liveness: *If all honest participants have been activated on ID and all associated messages have been delivered, then all honest participants have decided for ID.*

Integrity: *If all participants follow the protocol, and if some parties decide v for ID, then there is a participant who has proposed v for ID.*

Efficiency: *For every ID, the communication complexity of the instance ID is polynomially uniformly bounded.*

The protocol proposed by Cachin et al. [4] for the multi-valued validated Byzantine agreement relies on a threshold signature and a threshold coin-tossing protocol [5]. The expected message complexity is $O(n^2)$, and the expected communication complexity is $O(n^3 + n^2(K + |v|))$, where $|v|$ is the length of the value v which is proposed by any participant and K is the length of a threshold signature.

2.5 Pedersen VSS

Pedersen VSS [20] is information-theoretic secure, and it follows the following steps.

- The dealer chooses two random polynomials $f(z) = \sum_{i=0}^{t} a_j z^j$ and $f'(z) = \sum_{i=0}^{t} b_j z^j$ over $\mathbb{Z}_q$ such that $f(0) = \sigma$. The shares are $\sigma_i = f(i)$ and $\tau_i = f'(i)$. The shares (σ_i, τ_i) are sent to the participant P_i.
- The dealer commits to coefficients of the polynomials f and f' by publishing the values $A_j = g^{a_j} h^{b_j} \pmod p$ for $j = 0, 1, \ldots, t$. This will allow the participants to verify the received shares by testing

$$g^{\sigma_i} h^{\tau_i} \stackrel{?}{=} \prod_{j=0}^{t} (A_j)^{i^j} \pmod p, \text{where } i = 1, 2, \ldots, n. \quad (1)$$

- The reconstruction algorithm is the same as in the Shamir scheme [22].

3 Adaptive Proactive Secret Sharing without a Trusted Party

In this section, we construct our adaptive proactive secret sharing $(\mathcal{I}, \mathcal{U}, \mathcal{C}, \mathcal{R})$ without a trusted party (called `TPfree-APSS`), in which there is no trusted party. It would seem that the simplest way to do this is to run jointly a secret sharing based on the Feldman VSS [8], where each participant shares a random value $s_i \in Z_q$. However, in a asynchronous communication model, this is insecure since the Feldman VSS reveals $g^{s_i} \pmod p$, and hence the adversary might wait till he sees the g^{s_i} values corresponding to the random values of honest participants and use them to produce his shares. As the result, the secret $x \in Z_q$ might not be random. For example, it is easy to show that the adversary can control the last bit of the resultant secret x [11]. Our protocol is based on the Pedersen VSS [20], where the privacy of the secret is unconditionally secure, and it reveals nothing about g^{s_i} and g^x.

In the following subsections, we will describe the algorithms $\mathcal{I}$, $\mathcal{U}$, $\mathcal{C}$ and $\mathcal{R}$ respectively for our proposed `TPfree-APSS`. The proof of security of the proposed scheme will be given in Section 4.

3.1 Initialization Algorithm $\mathcal{I}$

In the initialization phase, every participant shares a random value s_i using Pedersen VSS [20]. After that, they agree on the participant contributions to the secret. This is done by the execution of the multi-valued validated Byzantine agreement protocol. Then participants compute their shares using the function we will define (see details below). This function can guarantee the robustness of our proposed adaptive proactive secret sharing which doesn't require a trusted party in the initialization phase. The detailed description of the algorithm $\mathcal{I}$ is presented as follows.

Algorithm $\mathcal{I}$:

Input: The security parameter q.

Output: The private outputs of P_i are secret shares x_i and the committed shares x'_i for $i = 1, 2, \ldots, n$. The public outputs are L and $V = \{V_0, V_1, \ldots, V_t\}$ (The sets L and V will be described in Remark 1 and in Step 5 respectively).

Steps: (executed by P_i)

1. P_i chooses two random polynomials $f_i(z) = \sum_{k=0}^{t} a_{ik} z^k$ and $f'_i(z) = \sum_{k=0}^{t} b_{ik} z^k$ of the degree t with $f_i(0) = s_i$, where s_i is a random number. Next P_i creates a set $A_i = \{A_{ik}\}$, where $A_{ik} = g^{a_{ik}} h^{b_{ik}} \pmod p$ for $k = 0, \ldots, t$. Further, P_i sends the message $(I, send, i, j, A_i, \sigma_{ij}, \tau_{ij})$ to each participant P_j, where $\sigma_{ij} = f_i(j)$ and $\tau_{ij} = f'_i(j)$ for $j \in \{1, 2, \ldots, n\}$.
2. P_i waits for a message $(I, send, j, i, A_j, \sigma_{ji}, \tau_{ji})$ from P_j, and verifies its correctness, then sends message $(I, echo, j, i, A_j)$ to every participant, which means that P_i has received a correct share from P_j.
3. Upon receiving $2t + 1$ valid *echo* messages $(I, echo, j, k, A_j)$ with the same A_j from different P_k, P_i puts all these $2t + 1$ *echo* messages into a set $\Pi_j = \{(I, echo, j, k, A_j)\}$ ($2t + 1$ *echo* messages mean that P_j has finished the distribution of the random value s_i). Upon getting $t + 1$ sets Π_j, P_i sets $L_i = \{(j, \Pi_j)\}$ of $(t + 1)$-tuples.
4. P_i proposes the set L_i for the multi-value validated Byzantine agreement proposed by Cachin et al. [4] with a tag ID/VBA. After deciding on some set L, P_i interpolates its share from L as $x_i = \sum\limits_{l \in L} \sigma_{li} \lambda_L^l$ and $x'_i = \sum\limits_{l \in L} \tau_{li} \lambda_L^l$, where $\lambda_L^l = \frac{l}{\prod\limits_{k \in L, k \neq l} (k - l)}$ is the normal Lagrange coefficient (if participant P_i has no σ_{li} for some $l \in L$, then other participants can reconstruct σ_{li} for P_i).
5. P_i computes $V_k = \prod\limits_{l \in L} (A_{lk})^{\lambda_L^l} \pmod p$, here $A_{lk} = g^{a_{lk}} h^{b_{lk}} \pmod p$ for $k = 0, 1, \ldots, t$. From Equation (2), the following equation holds $g^{x_i} h^{x'_i} = \prod\limits_{k=0}^{t} (V_k)^{i^k} \pmod p$. The sets L and $V = \{V_0, V_1, \ldots, V_t\}$ are the public outputs.

Remark 1. – In Step 2 of $\mathcal{I}$, the correctness of $(I, send, i, j, A_i, \sigma_{ij}, \tau_{ij})$ means that Equation (1) holds.

– In Step 4 of $\mathcal{I}$, there are two methods to reconstruct σ_{li} for P_i. One is by the Shamir method [22], the other is by the recovery method applied in Herzberg et al. work [12]. For the Shamir method, every P_j sends σ_{lj} to P_i, so that he can reconstruct the σ_{li} with Lagrange interpolation. With this method, a participant corrupted by the adversary can cheat that he has no σ_{li}, so that he can get the additional $f_l(\cdot)$. As the result, the adversary can control more than t participants. In this paper, we use the Herzberg et.al. recovery method to reconstruct σ_{li} for P_i.

– For Step 5 of $\mathcal{I}$, the following equation holds.

$$\begin{aligned} g^{x_i}h^{x'_i} &= g^{\sum\limits_{l\in L}\sigma_{li}\lambda_l} h^{\sum\limits_{l\in L}\tau_{li}\lambda_l} \\ &= \prod_{l\in L} g^{\sigma_{li}\lambda_l}h^{\tau_{li}\lambda_l} \\ &= \prod_{l\in L}(g^{\sum\limits_{k=0}^{t} a_{lk}i^k} h^{\sum\limits_{k=0}^{t} b_{lk}i^k})^{\lambda_l} \\ &= \prod_{l\in L}(\prod_{k=0}^{t}(A_{lk})^{i^k})^{\lambda_l} \\ &= \prod_{k=0}^{t}(\prod_{l\in L}(A_{lk})^{\lambda_l})^{i^k} \\ &= \prod_{k=0}^{t}(V_k)^{i^k} \pmod p. \end{aligned} \tag{2}$$

– For Step 4 of $\mathcal{I}$, L is the set decided by multi-value validated Byzantine agreement. From the Integrity property of Definition 2, L is some L_i proposed by P_i.

3.2 Update Algorithm $\mathcal{U}$

In the update phase, the threshold t and the number n of participants are fixed. The corrupted participant in the previous interval can be reset or replaced by a new participant. The difference between $\mathcal{U}$ and $\mathcal{I}$ is that the participants in $\mathcal{U}$ has to reshare their shares from the previous interval instead of sharing a random number in the algorithm $\mathcal{I}$.

The detailed description of the algorithm $\mathcal{U}$ is presented as follows.

Algorithm $\mathcal{U}$:

Input: x_i, x'_i – the private inputs of P_i; L and $V = \{V_0, V_1, \ldots, V_t\}$– the public inputs.
Output: $\bar{x}_i$, $\bar{x}'_i$ – the private output of P_i. The sequence $(\bar{x}_1, \ldots, \bar{x}_n)$ represents shares of the same secret x and their commitments $(\bar{x}'_1, \ldots, \bar{x}'_n)$ for the current interval. The public outputs are $\bar{L}$ and $\bar{V}$.

Steps:(executed by the participant P_i)
1. P_i chooses two random polynomials $\bar{f}_i(z) = \sum_{k=0}^{t} \bar{a}_{ik}z^k$ and $\bar{f}'_i(z) = \sum_{k=0}^{t} \bar{b}_{ik}z^k$ of t degree such that $\bar{f}_i(0) = x_i$ and $\bar{f}'_i(0) = x'_i$. He calculates $\bar{\sigma}_{ij} = \bar{f}_i(j)$, $\bar{\tau}_{ij} = \bar{f}'_i(j)$ for $j \in \{1, \ldots, n\}$ and $\bar{A}_i = \{\bar{A}_{ik}\}$, where $\bar{A}_{ik} = g^{\bar{a}_{ik}}h^{\bar{b}_{ik}} \pmod p$ for $k = 0, 1, \ldots, t$. After P_i distributes the shares, then he immediately erases his shares and the polynomials $\bar{f}_i(z)$ and $\bar{f}'_i(z)$.
2. P_i waits for a message $(U, send, j, i, \bar{A}_j, \bar{\sigma}_{ji}, \bar{\tau}_{ji})$ from P_j, verifies its correctness, then sends message $(U, echo, j, i, \bar{A}_j)$ to every participant, which means that P_i has received a correct share from P_j.
3. P_i waits for $2t+1$ *echo* messages $(U, echo, j, k, \bar{A}_j)$ with the same $\bar{A}_j$ from different P_k. P_i puts all these $2t+1$ *echo* messages into a set $\bar{\Pi}_j = \{(U, echo, j, k, \bar{A}_j)\}$. Upon getting $t+1$ such $\bar{\Pi}_j$, P_i sets $\bar{L}_i = \{(j, \bar{\Pi}_j)\}$ of $(t+1)$-tuples.
4. P_i proposes the set $\bar{L}_i$ for multi-value validated Byzantine agreement with ID/BVA. After deciding a set $\bar{L}$, then P_i interpolates his share from $\bar{L}$ using the following equations $\bar{x}_i = \sum\limits_{l\in\bar{L}} \bar{\sigma}_{li}\lambda^l_{\bar{L}}$ and $\bar{x}'_i = \sum\limits_{l\in\bar{L}} \bar{\tau}_{li}\lambda^l_{\bar{L}}$. At the meanwhile, P_i computes $\bar{V}_k = \prod\limits_{l\in\bar{L}}(\bar{A}_{lk})^{\lambda^l_{\bar{L}}} \pmod p$ for $k = 0, 1, \ldots, t$. $\bar{L}$ and $\bar{V} = \{\bar{V}_0, \bar{V}_1, \ldots, \bar{V}_t\}$ are the public outputs.

Remark 2. In Step 2 of $\mathcal{U}$, the correctness of $(U, send, i, j, A_i, \sigma_{ij}, \tau_{ij})$ means that not only Equation (1) holds but also $\bar{A}_{i0} = \prod_{j=0}^{t}(V_i)^{i^j} \pmod p$. So there is $g^{x_i}h^{x'_i} = \bar{A}_{i0} \pmod p$, which means that P_i distributes the share he has owned in the previous interval.

3.3 Change Threshold Algorithm $\mathcal{C}$

We denote the participants set in the previous interval as S, the one in the current interval as T, here $|S| = n$ and $|T| = n'$. As we consider the change of the threshold from t to t', an intuitive method is that the participants reshare their shares using a polynomial of degree t'. The change threshold algorithm is somehow similar to the update algorithm. The difference is that every $P_i \in S$ distributes his share to every participant Q_j in T. After that, every Q_j agrees on the participant contribution to the new shares. The contribution is completely determined by the parameter t, so we need to consider the change of t only. For the clarity of presentation, the detailed description of the algorithm $\mathcal{C}$ is given below.

Algorithm $\mathcal{C}$: (x_i, x_i', L and V are the same as in the algorithm $\mathcal{U}$ which are from the previous interval)

Input: x_i, x_i' – the private inputs of $P_i \in S$; L and $V = \{V_0, \ldots, V_t\}$ – the public inputs of $P_i \in S$.

Output: $\bar{x}_j$, $\bar{x}_j'$ – the private outputs of $P_j \in T$. $(\bar{x}_1, \ldots, \bar{x}_{n'})$ are shares of the same secret and $(\bar{x}_1', \ldots, \bar{x}_{n'}')$ are their commitments for the current interval. The public outputs are $\bar{L}$ and $\bar{V} = \{\bar{V}_0, \bar{V}_1, \ldots, \bar{V}_{t'}\}$.

Steps :

1. $P_i \in S$ designs two random polynomials $\bar{f}_i(z) = \sum_{k=0}^{t'} \bar{a}_{ik} z^k$ and $\bar{f}_i'(z) = \sum_{k=0}^{t'} \bar{b}_{ik} z^k$ of the degree t' such that $\bar{f}_i(0) = x_i$ and $\bar{f}_i'(0) = x_i'$. He calculates $\bar{\sigma}_{ij} = \bar{f}_i(j)$, $\bar{\tau}_{ij} = \bar{f}_i'(j)$ and $\bar{A}_i = \{\bar{A}_{ik}\}$, where $\bar{A}_{ik} = g^{\bar{a}_{ik}} h^{\bar{b}_{ik}} \pmod{p}$ for $k = 0, 1, \ldots, t'$.
   P_i distributes his share by sending message $(C, \mathit{send}, i, j, \bar{A}_i, \bar{\sigma}_{ij}, \bar{\tau}_{ij})$ to every $Q_j \in T$. After P_i distributes the shares, then he immediately erases his shares and the polynomials $\bar{f}_i(z)$ and $\bar{f}_i'(z)$.
2. $Q_j \in T$ receives the *send* message from P_i and checks its correctness. If the *send* message is correct, then Q_j sends $(C, \mathit{echo}, i, j, \bar{A}_i)$ to every $Q_k \in T$.
3. $Q_j \in T$ waits for $2t' + 1$ *echo* messages $(C, \mathit{echo}, i, k, \bar{A}_i)$ with the same $\bar{A}_i$ from different Q_k. Q_j puts all these $2t' + 1$ *echo* messages into a set $\bar{\Pi}_i = \{(C, \mathit{echo}, i, k, \bar{A}_i)\}$. Upon getting $t + 1$ such $\bar{\Pi}_i$, P_j sets $\bar{L}_j = \{(i, \bar{\Pi}_i)\}$ of $(t + 1)$-tuples.
4. $Q_j \in T$ proposes the set $\bar{L}_j$ for multi-valued validated Byzantine agreement with ID/VBA and decides on a $t + 1$ tuple set $\bar{L}$. After that, Q_j interpolates his share from $\bar{L}$ using the following equations $\bar{x}_j = \sum_{i \in \bar{L}} \bar{\sigma}_{ij} \lambda_{\bar{L}}^i$ and $\bar{x}_j' = \sum_{i \in \bar{L}} \bar{\tau}_{ij} \lambda_{\bar{L}}^i$. At the meanwhile, he can get $\bar{V} = \{\bar{V}_0, \bar{V}_1, \ldots, \bar{V}_{t'}\}$ with the same way as in the algorithm $\mathcal{U}$.

Remark 3. – In Step 2 of $\mathcal{C}$, the verification of the message correctness is the same as for the algorithm $\mathcal{U}$.
- In Step 3 of $\mathcal{C}$, because $n' \geq 3t' + 1$ in the current interval, $Q_j \in T$ can get $2t' + 1$ *echo* messages for participant $P_i \in S$, that means each $P_i \in S$ can eventually distribute his share.
- In Step 3 of $\mathcal{C}$, because $n \geq 3t + 1$ in S, $Q_j \in T$ can get $t + 1$ $\bar{\Pi}_i$, that means participants in T can agree with a $(t + 1)$-tuple set in Step 4.
- We can deal with both the decreasing and increasing of the threshold value in algorithm $\mathcal{C}$, because of the use of resharing technique.

3.4 Reconstruction Algorithm $\mathcal{R}$

In each interval, $t + 1$ honest participants can reconstruct the secret. The detailed algorithm $\mathcal{R}$ is presented as follows.

Algorithm $\mathcal{R}$:

Input: x_i – the private inputs of P_i; V – the public input.
Output: Public output of P_i is the secret x.

Steps:(executed by the participant P_i)

```
1. $P_i$ sends the message $(recons, x_i, x'_i)$ to $P_j$ for $j \in \{1, \ldots, n\}$.
2. Upon receiving a message $(recons, x_k, x'_k)$ from $P_k$, $P_i$ verifies this message by checking if $g^{x_i} h^{x'_i} \overset{?}{=} \prod_{j=0}^{t} (V_i)^{i^j} \pmod{p}$.
3. Upon receiving $(t+1)$ valid $recons$ messages from different $t+1$ participants, $P_i$ can reconstruct $x$ with Lagrange interpolation by using these $(t+1)$ messages .
```

4 Security Proof

From the Definition 1, in order to prove the security of `TPfree-APSS`, we have to prove that `TPfree-APSS` satisfies three properties: robustness, privacy and liveness.

4.1 Proof of Robustness

From the Definition 1, we have to prove that the collection of $t+1$ participants in any interval can always reconstructs the secret x. There are three cases to discuss, namely the robustness of the algorithms $\mathcal{I}$, $\mathcal{U}$ and $\mathcal{C}$ respectively.

Lemma 1. *Let $n \geq 3t+1$, then the algorithm $\mathcal{I}$ is robust under the mobile Byzantine adversary model.*

Proof. In order to prove that $\mathcal{I}$ is robust, we have to show that every collection of $t+1$ or more participants reconstructs the same secret.

Let L be a currently active set of participants agreed in Step 4 of the algorithm $\mathcal{I}$, L' and L'' be two random reconstruction sets with $t+1$ participants, and x', x'' be the reconstructed values from L' and L'' respectively. Then the following equation holds.

$$x' = \sum_{k \in L'} s_k \lambda_{L'}^k = \sum_{k \in L'} \sum_{l \in L} \sigma_{lk} \lambda_L^l \lambda_{L'}^k = \sum_{l \in L} \sum_{k \in L'} \sigma_{lk} \lambda_{L'}^k \lambda_L^l = \sum_{l \in L} s_l \lambda_L^l. \tag{3}$$

For the same reason, we have $x'' = \sum_{k \in L''} s_k \lambda_{L''}^k = \sum_{l \in L} s_l \lambda_L^l$. Then the equation $x' = x''$ holds, which means that the robustness property of $\mathcal{I}$ holds. □

From now on, we denote the secret as x_0.

Lemma 2. *Let $n \geq 3t+1$, then the algorithms $\mathcal{U}$ and $\mathcal{C}$ are robust under the mobile Byzantine adversary model.*

Proof. In order to prove robustness property of the algorithm $\mathcal{U}$, we have to demonstrate that the new secret computed at the end of the update phase should be the same as the secret x_0, which means any subset of $t+1$ of the new shares can reconstruct the secret x_0. Assume that in the current phase, the reconstructed secret is $\bar{x}_0$. Let $\bar{L}$ be the agreed set in the current interval and K be a random set which contains $t+1$ participants. Then

$$\bar{x}_0 = \sum_{k \in K} \bar{x}_k \lambda_K^k = \sum_{k \in K} \sum_{l \in \bar{L}} \bar{\sigma}_{lk} \lambda_{\bar{L}}^l \lambda_K^k = \sum_{l \in \bar{L}} \sum_{k \in K} \bar{\sigma}_{lk} \lambda_K^k \lambda_{\bar{L}}^l = \sum_{l \in \bar{L}} x_l \lambda_{\bar{L}}^l = x_0. \tag{4}$$

Thus, the robustness property of U holds.

The robustness of the algorithm $\mathcal{C}$ is similar with that in the algorithm $\mathcal{U}$. The difference is that in the threshold change phase, with the threshold changes from t to t', any subset of $t'+1$ of the new shares can reconstruct the secret x_0.

Assume that the reconstructed secret is $\bar{x}'_0$. Let $\bar{L}'$ be the agreed set in the current interval with $t+1$ participants (notice that the participants in the $\bar{L}'$ come from the previous interval) and K' be a random set which contains $t'+1$ participants. Then

$$\bar{x}'_0 = \sum_{k\in K'} \bar{x}_k \lambda^k_{K'} = \sum_{k\in K'}\sum_{l\in \bar{L}'} \bar{\sigma}_{lk}\lambda^l_{\bar{L}'}\lambda^k_{K'} = \sum_{l\in \bar{L}'}\sum_{k\in K'} \bar{\sigma}_{lk}\lambda^k_{K'}\lambda^l_{\bar{L}'} = \sum_{l\in \bar{L}'} x_l \lambda^l_{\bar{L}'} = x_0. \tag{5}$$

Thus, the robustness property of algorithm $\mathcal{C}$ holds. □

Theorem 1. *Under the mobile Byzantine adversary model, `TPfree-APSS` satisfies the robust property.*

Proof. The proof of the above theorem can be guaranteed from Lemmas 1 and 2. □

4.2 Proof of Privacy

For the privacy of `TPfree-APSS`, there are two cases. One is the privacy of the algorithm $\mathcal{I}$ which means that it reveals no information about secret in a fixed interval. The other case is that the transition to the new interval does not leak any information to the adversary about the secret, by using either algorithm $\mathcal{U}$ or algorithm $\mathcal{C}$.

Lemma 3. *Assuming the hardness of the discrete-logarithm problem, the algorithm $\mathcal{I}$ of `TPfree-APSS` reveals no information about the secret under the adversarial model with Byzantine faults and $n \geq 3t+1$, where t is the number of dishonest participants.*

Proof. This lemma is proven in the Appendix A. □

Theorem 2. *Assuming the hardness of the discrete-logarithm problem, `TPfree-APSS` preserves privacy under the adversarial model with Byzantine faults with $n \geq 3t+1$.*

Proof. We proceed by induction. The privacy of the initialization is guaranteed by the Lemma 3. Assume that at each time interval $r = 1, 2, \ldots, e-1$ the theorem holds. It means that after the update or threshold change phase of time interval $e-1$, the adversary has known no information about the secret.

Let T_1 be the set of t_1 servers that the adversary corrupted into interval $e-1$, T_2 be the set of t_2 servers that the adversary corrupted into in interval e. We also denote by S_1 and S_2 the set of shares corresponding to the servers in T_1 and T_2, respectively. We assume that the threshold changes from t to t' when the time interval changes from $e-1$ to e. So that, we have $t_1 \leq t$ and $t_2 \leq t'$. Here, we assume that $t_1 = t$ and $t_2 = t'$ (for $t_1 < t$ and $t_2 < t'$, the adversary knows less about the secret). Further, we assume that the threshold $t \geq t'$ and then $T_2 \subseteq T_1$.

We now show that the availability of all this information about shares and threshold change does not provide information about x_0. Note that the update case is the special case of the threshold change.

Since we assume that the t shares from interval $e-1$ are known (from the view of the adversary), fixing the secret x_0 determines the interpolation polynomial $f^{(e-1)}$ of degree t, corresponding to interval $e-1$. Similarly, from the t' known shares of interval e, fixing the secret x_0 determines the polynomial f^e of degree t'. The difference between $f^{(e-1)}$ and $f^{(e)}$ represents a t-degree polynomial with free coefficient zero and its evaluation on the points corresponding to the servers in T_2 is consistent with the shares available to the adversary.

Since the above argument holds for any value of x_0, then all possible values of x_0 are consistent with the available information to the adversary. In other words, no information on x_0 is revealed. □

4.3 Proof of Liveness

Theorem 3. *Let $n \geq 3t+1$, `TPfree-APSS` preserves the liveness under the mobile Byzantine adversary model.*

Proof. In the following, we analyze the liveness (termination) of the algorithms $\mathcal{I}$,$\mathcal{U}$, $\mathcal{C}$, respectively.

- Liveness of algorithm $\mathcal{I}$ depends on termination of Step 3 and 4. Because there are at most t corrupted participants in the interval, each participant waits for $2t+1$ *echo* messages for participant P_i from different participants, which confirms that the P_i has completed the distribution their share. The *echo* messages are stored in the set Π_i. Because there are at least $t+1$ honest participant, every participant can get $t+1$ Π_i, so Step 3 terminates. The termination of Step 4 is guaranteed by the termination of the multi-valued Byzantine agreement protocol.
- The proof of the liveness of algorithm $\mathcal{U}$ is the same as in the case of the algorithm $\mathcal{I}$.
- Because the number of corrupted participants in the current interval is at most t' and $n' \geq 3t'+1$, every $Q_j \in T$ can wait for $2t'+1$ *echo* messages for evert honest $P_i \in S$, which means every honest participant in S has finished the resharing of his share. As the threshold value in S is t, there are at least $2t+1$ participants would like to distribute their shares, then every participant $Q_j \in T$ will eventually get $t+1$ $\bar{\Pi}_i$ in each $\bar{L}_j$, which means the termination of Step 3 of the algorithm $\mathcal{C}$. The termination of Step 4 is guaranteed by the multi-valued Byzantine agreement protocol. Finally, every participant in T can compute their new shares according to a polynomial of degree t'.

□

4.4 Security of Proposed Scheme

Theorem 4. *Assuming the hardness of the discrete-logarithm problem, $n \geq 3t+1$ and the threshold change from t to t' and the number of participants change from $n \geq 3t+1$ to $n' \geq 3t'+1$, respectively, `TPfree-APSS` is a secure adaptive proactive secret sharing under the mobile Byzantine adversary model.*

Proof. It can be obtained from Theorems 1, 2 and 3. □

4.5 Discussion of Complexity

We consider the complexity of `TPfree-APSS` from two aspects, one is message complexity, the other is communication complexity. It can be seen that the complexity of algorithms $\mathcal{I}$ and $\mathcal{U}$ are the same, and algorithm $\mathcal{U}$ is a special case of algorithm $\mathcal{C}$, so we only analysis the complexity of $\mathcal{C}$.

First, we consider the message complexity of $\mathcal{C}$. In the Step 1, each participant $P_i \in S$ send *send* message to every participant $Q_j \in T$. So, there are $n'n$ messages in Step 1. In the Step 2, each participant $Q_j \in T$ send *echo* message for each $P_j \in S$ to every participant in T. So, there are $(n')^2 * n$ messages in Step 2. In Step 4, the message complexity, which is dominated by the a multi-valued validated Byzantine agreement [5], is $O((n')^2)$. As a result, the message complexity of $\mathcal{C}$ is $O((n')^2n)$.

Now we analyze the communication complexity of $\mathcal{C}$. The length of *send*, *echo* messages, which is dominated by A_i, is $O(t')$, so the communication complexity of the Step 1 and Step 2 are $O(n'nt')$ and $O((n')^2nt')$ respectively. In Step 4, the communication complexity, which is dominated by the multi-valued validated Byzantine agreement [5], is $O(n^3+n^2(|\bar{L}|))$, here the $\bar{L}$

is composed of $t+1$ $\bar{\Pi}_i$ and each $\bar{\Pi}_i$ is composed of $2t'+1$ *echo* messages. So, the communication complexity of Step 4 is $O((n')^3 + (n')^2t(t')^2)$. Therefore, the communication complexity of $\mathcal{C}$ is $O((n')^3+(n')^2t(t')^2+(n')^2nt')$. Furthermore, we can do some optimization to the communication complexity by using the digest of $\bar{A}_i$ instead of $\bar{A}_i$ itself in the *echo* message.

In order to simple the formal of the complexity, we assume n and n', t and t' are in the same level in complexity. Thus, `TPfree-APSS` has message complexity $O(n^3)$ and communication complexity $O((n)^3t + (n)^2t^3)$.

5 Conclusion

In this paper, we design a novel adaptive proactive secret sharing without a trusted party in asynchronous communication system. Using the resharing technology and multi-valued Byzantine agreement protocol, the proposed scheme is secure and practical with the message complexity $O(n^3)$ and communication complexity $O((n)^3t + (n)^2t^3)$.

References

1. Alon, N., Galil Z., Yung, M.: Dynamic-resharing verifiable secret sharing against mobile adversary. In: Spirakis, P. (ed.) 3rd European Symp. On Algorithms (ESA'95). LNCS, vol. 979, pp. 523–537. Springer, Heidelberg (1995)
2. Blakley, G. R.: Safeguarding cryptographic keys. In: Proc. of AFIPS National Computer Conference. vol. 48, pp. 313–317 (1979)
3. Cachin, C., Kursawe, K., Lysyanskaya, A., Strobl, R: Asynchronous verifiable secret sharing and proactive cryptosystem. In: Proc. 9th ACM Conference on Computer and Communications Security, pp. 88–97 (2002)
4. Cachin, C., Kursawe, K., Petzold, F., Shoup, V.: Secure and efficient asynchronous broadcast protocols (extended abstract). In: Kilian, J. (ed.) CRPYTO'01. LNCS, vol. 2139, pp. 524–541, Springer, Heidelberg (2001)
5. Cachin, C., Kursawe, K., Shoup, V.: Random oracles in constantinople: Practical asynchronous Byzantine agreement using cryptography. In: 19th ACM Symp. on Principles of Distributed Computing, pp. 123–132 (2000)
6. Canetti, R., Herzberg, A.: Maintaining security in the presence of transient faults. In: Desmedt, Y. (ed.) CRYPTO'94. LNCS, vol. 839, pp. 425–438. Springer, Heidelberg (1994)
7. Canetti, R., Rabin, T.: Fast asynchronous Byzantine agreement with optimal residence. In: Proc. 25th Annual ACM Symposium on Theory of Computing, pp. 42–51 (1993)
8. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: Proc. 28th Annual Symposium on the Foundations of Computer Science, pages 427–437 (1987)
9. Fischer, M. J., Lynch, N. A., Paterson, M. S.: Impossibility of distributed consensus with one faulty process. J. ACM 32(2), 374–382 (1985)
10. Frankel, Y. Gemmell, P., MacKenzie, P., Yung, M.: Proactive RSA. In: Burton, S., Kaliski., Jr. (eds.) CRYPTO'97. LNCS, vol. 1294, pp. 440–454. Springer, Heidelberg (1997)
11. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation in discrete-log based cryptosystems. J. Cryptology 20 (1), 51–83 (2007)
12. Herzberg, A., Jakobsson, M., Jarecki, S., Krawczyk, H., Yung, M.: Proactive public key and signature systems. In: Proc. of the 1997 ACM Conference on Computers and Communication Security, pp. 100–110 (1997)
13. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: How to cope with perpetual leakage. In: Coppersmith, D. (ed.) CRYPTO'95. LNCS, vol. 963, pp. 339–352. Springer, Heidelberg (1995)
14. Ito, M., Saito, A., Nishizeki, T.: Secret sharing scheme realizing general access structure. In: Proc. of IEEE Global Communication Conference, pp. 99–102 (1998)
15. Lamport, L., Shostak, R., Pease. M.: The Byzantine generals problem. ACM transaction on Programming language and system 4 (3), 382–401 (1982)
16. Nikov, V., Nikova, S.: On proactive secret sharing scheme. In: Handschuh, H., Anwar Hasan, M. (eds.) SAC'04. LNCS, vol. 3357, pp. 308–325. Springer, Heidelberg (2005)
17. Ostrovsky, R., Yung, M.: How to withstand mobile virus attack. In: PODC'1991, pp. 51–59 (1991)
18. Pease, M., Shostak, R., Lamport, L.: Reaching agreement in the presence of faults. J. Association for Computing Machinery 27 (2), 228–234 (1980)
19. Pedersen, T.: A threshold cryptosystem without a trusted party. In: Davies, D. (ed.) EUROCRYPT'91. LNCS, vol. 547, pp. 129–140. Springer, Heidelberg (1991)

20. Pedersen, T.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
21. Schultz, D., Liskov, B., Liskov, M.: MPSS: Mobile proactive secret sharing. In: 27th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, pp. 458–458 (2008)
22. Shamir, A.: How to share a secret. Comm of ACM. 22, 612–613 (1979)
23. Stinson, D. R., Wei, R.: Unconditionally secure proactive secret sharing scheme with combinatorial structures. In: Heys, H. M., Adams, C. M. (eds.) SAC'99. LNCS. vol. 1758, pp. 200–214. Springer, Heidelberg (2000)
24. Wang, S., Tsai, Y., Chen, P.: Strategies of proactive (k, n) threshold secret sharing and application in a secure message exchange system. J. Computers 19 (1), 29–38 (2008)
25. Zhou, L., Schneider, F. B., Ranesse, R. V.: APSS: Proactive secret sharing in asynchronous systems. ACM Transaction on Information and System Security 8 (3), 259–286 (2005)

A Proof of Lemma 3

Proof. We recall that the privacy property is if an adversary who knows no more than t shares knows nothing about the secret.

In order to prove that the adversary who has no more than t shares knows nothing about the secret x_0, we have to show that for every value $\tilde{x}_0$, there exist two polynomials $\tilde{f}, \tilde{f}' \in \mathbb{Z}_q[x]$ of degree t that are consistent with the adversary view and such that $\tilde{f}(0) = \tilde{x}_0$.

Fix any point in time, let B be the index set of participants that are corrupted. Here, we assume $|B| = t$ (for $|B| < t$, the adversary knows less). Without loss of generality, we assume that the set B contains the first t participants. The adversary view consists of the polynomials $f_i(z)$, $f_i'(z)$, and the shares x_i, x_i' for $i \in B$, and all the commitments A_j for $j \in \{1, \dots, n\}$.

Note that, for any $\tilde{x}_0$, there is a unique value $\tilde{x}_0'$ such that $g^{\tilde{x}_0} h^{\tilde{x}_0'} = g^{x_0} h^{x_0'}$. For $i \in B$, the values $\tilde{x}_0$ and $\tilde{x}_0'$ together with x_i and x_i', define uniquely two polynomials $\tilde{f}_0$, $\tilde{f}_0' \in \mathbb{Z}_q[x]$ of degree t such that $\tilde{f}_0(0) = \tilde{x}_0$ and $\tilde{f}_0'(0) = \tilde{x}_0'$, as well as

$$\tilde{f}_i(z) = f_i(z), \tilde{f}_i'(z) = f_i'(z). \tag{6}$$

It remains to show that $A_{ik} = g^{\tilde{a}_{ik}} h^{\tilde{b}_{ik}}$ for $k \in \{0, \dots, t\}$ and $i \in \{0, 1, \dots, n\}$. Define $e_i(z) = f_i(z) + l f_i'(z) = \sum_{k=0}^{t} e_{ik} z^k$ and $\tilde{e}_i(z) = \tilde{f}_i(z) + l \tilde{f}_i'(z) = \sum_{k=0}^{t} \tilde{e}_{ik} z^k$, where $l = \log_g h$, then we have $A_{ik} = g^{e_{ik}} = g^{a_{ik} + \log_g h \times b_{ik}} = g^{a_{ik}} h^{b_{ik}} \pmod p$. From Equation (6), we have $A_{ik} = g^{\tilde{a}_{ik}} h^{\tilde{b}_{ik}} \pmod p$ for $k \in \{0, \dots, t\}$ and $i \in B$. It remains to show that $e_i(z) = \tilde{e}_i(z)$ for $i \in \{0\} \cup \{t + 1, \dots, n\}$.

For $i = 0$, we have

$$g^{\tilde{e}_0(0)} = g^{\tilde{f}_0(0) + l \tilde{f}_0'(0)} = g^{\tilde{x}_0} h^{\tilde{x}_0'} = g^{x_0} h^{x_0'} = g^{f_0(0) + l f_0'(0)} = g^{e_0(0)},$$

and this implies $e_0(0) = \tilde{e}_0(0)$. For $\tilde{f}_0(i) = x_i = f_0(i)$ and $\tilde{f}_0'(i) = x_i' = f_0'(i)$, we can get that $e_0(z) = \tilde{e}_0(z)$.

Assume that $L = B \cup \{P_{t+1}\}$. For $i = t + 1$, define $\tilde{f}_{t+1}(z) = \Sigma_{i \in \{0\} \cup B} \tilde{f}_i(z) \lambda^i$ and $\tilde{f}_{t+1}'(z) = \Sigma_{i \in \{0\} \cup B} \tilde{f}_i'(z) \lambda^i$. From the algorithm $\mathcal{I}$, we have $f_{t+1}(z) = \Sigma_{i \in \{0\} \cup B} f_i(z) \lambda^i$ and $f_{t+1}'(z) = \Sigma_{i \in \{0\} \cup B} f_i'(z) \lambda^i$. So

$$\begin{aligned} g^{\tilde{e}_{t+1}(0)} &= g^{\tilde{f}_0(0)\lambda^0 + l\tilde{f}_0'(0)\lambda^0 + (\Sigma_{i\in B}\tilde{f}_i(0)\lambda^i + l\tilde{f}_i'(0)\lambda^i)} \\ &= g^{\tilde{e}_0(0)\lambda^0 + \Sigma_{i\in B}\tilde{e}_i(0)\lambda^i} \\ &= g^{e_0(0)\lambda^0 + \Sigma_{i\in B} e_i(0)\lambda^i} \\ &= g^{f_0(0)\lambda^0 + l f_0'(0)\lambda^0 + (\Sigma_{i\in B} f_i(0)\lambda^i + l f_i'(0)\lambda^i)} \\ &= g^{e_{t+1}(0)}. \end{aligned} \tag{7}$$

For the same reason, we can get $g^{\tilde{e}_{t+1}(i)} = g^{e_{t+1}(i)}$ for $i = 1, \dots, t$. So $\tilde{e}_{t+1}(z) = e_{t+1}(z)$.

For $i = t + 2, \dots, n$, because $P_i \in \{1, \dots, n\} \setminus B$ and it doesn't contribute to the secret x_0 from the view of the adversary, we can choose random polynomials $\tilde{f}_i(z)$ and $\tilde{f}_i'(z)$ such that $A_{ik} = g^{\tilde{a}_{ik}} h^{\tilde{b}_{ik}} \pmod p$. □