An Unconditionally Secure Social Secret Sharing Scheme

Mehrdad Nojoumian, Douglas R. Stinson, and Morgan Grainger

David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, ON, N2L 3G1, Canada
{mnojoumi, dstinson, mjgrainger}@uwaterloo.ca

Abstract. We introduce the notion of a *Social Secret Sharing Scheme*, in which shares are allocated based on a player's reliability and the way he interacts with other participants. During the share refresh phase, weights of participants are adjusted in a way that participants who cooperate will end up with more shares than those who defect. On the other hand, corrupted players will be disenrolled immediately for the computation safety. Our motivation is that, in real world applications, components of a secure multiparty computation framework may have different levels of importance as well as credibility. Therefore, a robust construction should balance these two factors respectively, that is adjusting the responsibility based on the reliability. The proposed construction has a variety of desirable properties. It is an unconditionally verifiable scheme in the sense that it can detect malicious participants without relying on any computational assumptions. The scheme proactively renews shares at each cycle without changing the secret, and allows trusted participants to gain more authority in the scheme, i.e., a dynamic access structure. The other prominent property of the scheme is that, it gradually reduces the influence of irresponsible players due to the self-reinforcement property of social interactions among players.

1 Introduction

The growth of Internet has created amazing opportunities for *secure multiparty computations* where various users, intelligent agents, or computer servers cooperate in order to conduct computation tasks based on the private data they each provide [10]. Since these computations could be among untrusted participants or competitors, therefore, the privacy of each participant's input is an important factor.

As mentioned in the literature [11], a fundamental method used in secure multiparty computations is the *secret sharing scheme* [20, 5], where a secret divided into different shares for distribution among participants (private data), and a subset of participants then cooperate in order to reveal the secret (computation result). In particular, Shamir proposed the (t, n)*-threshold secret sharing scheme*, in which the secret is divided into n shares for distribution among players. The shares are constructed such that any t participants can combine their shares to reveal the secret, but any set of $t-1$ participants cannot learn anything about the secret.

Sample applications of such schemes are: *joint signature or decryption*, where a group of participants sign documents or decrypt messages with the intention that only if all of them or a subset of participants cooperate then a signature or a message can be generated [11], *shared RSA keys*, in which a number of players collaborate to jointly construct an RSA key [7], *electronic auctions with private bids*, where a group of agents perform sealed-bid electronic auctions while preserving the privacy of the submitted bids [12].

1.1 Motivation

Our motivation is that, in real world applications, components of a secure multiparty computation framework may have different *levels of importance*, i.e., the number of shares a player has, as well as *credibility*, i.e., cooperation with other players for the secret recovery. Therefore, a robust construction should balance these two factors respectively, that is adjusting the responsibility based on the reliability. Assume a major shareholder has been attacked, if the scheme is not re-arranged the security cost would be severe. On the other hand, if a player with a small number of shares is working reliably for some period of time, it might be rational to assign him more shares. Although our goal is to focus on the theoretical aspects of such a construction, we motivate the proposed scheme by the following scenario.

Suppose shares of a secret have been distributed among various servers based on their weights in a secure computing environment, consequently, revealing the secret will trigger an action. Three possible

behaviors can be considered for each player. *Cooperation*: the server is available at the secret recovery time and sends correct shares. *Defection*: the server is not available at the time of recovery or probably responds with delay. *Corruption*: the server has been compromised by an active adversary and sends wrong shares to prevent the recovery process. The aim is to monitor participants' behavior over time to regulate servers' responsibility. As an example of this scenario, we can refer to real time systems that are subject to operational deadlines.

1.2 Our Contributions

The contributions of this paper are as follows: First, we modify the existing verifiable and proactive secret sharing model [23, 9] to construct a novel secret sharing scheme with dynamic weighting based on the players' credibility analysis over time, i.e., having different number of shares in various time intervals after the initial setting. In the literature, there exist dynamic schemes with different properties than our construction, such as schemes in which one can activate various access structures [6], enroll or disenroll participants [27], or change the threshold [24]. Second, we construct an unconditionally secure dealer-free protocol for enrolling new participants in a threshold secret sharing scheme. Third, we propose required techniques for the weight escalation and/or reduction based on the existing trust computation model [17].

In our construction, each player is assigned an initial weight, which then is adjusted based on his behaviors in subsequent periods. Participants who cooperate will end up with more shares than those who defect. On the other hand, corrupted players will be disenrolled immediately for the computation safety. The reason for a corruption might be an active attack or a computational failure, therefore, the corrupted server is able to come back to the scheme only after being fixed, and will be treated as a newcomer. The scheme is called a *social secret sharing scheme* since it can be visualized in terms of players collaborating to recover the secret in a social network based on their reputations; similar to human social life in which people share more secrets with whom they really trust and vice versa.

1.3 Organization

This paper is organized as follows. Section 2 provides some background on different types of secret sharing schemes. Section 3 illustrates the proposed constructions in details. Section 4 demonstrates required techniques for weight tuning based on players' past behaviors. Section 5 proves the security of the constructions. Finally, Section 6 outlines concluding remarks and future work.

2 Preliminaries

2.1 Threshold Schemes

Given a set of participants $\mathcal{P}$, we want to protect a secret so that some participants can recover the secret, while other subsets cannot. If we let $2^{\mathcal{P}}$ denote the set of subsets of $\mathcal{P}$, then we can define $\Gamma \subseteq 2^{\mathcal{P}}$ to be the subsets of users who are authorized to access the secret; we call Γ an access structure [21]. A perfect secret sharing scheme realizing Γ has the property that, for any $A \subseteq \mathcal{P}$: (1) If $A \subseteq \Gamma$, then A can recover the secret, in this case, A is an *authorized subset* of Γ. (2) If $A \not\subseteq \Gamma$, then A can learn nothing about the secret. In a (t, n)-*threshold secret sharing scheme*, Γ has the property that for all $A \subseteq P$, $A \in \Gamma$ if and only if $|A|$ is equal or larger than some threshold value t. Intuitively, threshold schemes require action on the part of some number of participants to retrieve the secret. This idea of distribution of responsibility is widely used in practice, and motivated the first secret sharing schemes [20].

2.2 Verifiable Schemes

In *verifiable secret sharing schemes*, participants can check that their shares are consistent with those of the other participants in the system. Unconditionally secure verifiable secret sharing schemes must satisfy

the following properties: the information used to verify shares must not reveal any information about the secret, and the shares must be consistent if and only if every user verifies their share successfully. The first complete definition of verifiability is given by Chor et al. [8], but their solution assumes honest participants and makes computational assumptions. Tompa and Woll [26] consider the case of dishonest participants, but assume an honest dealer.

Ben-Or et al. [2] illustrate that no unconditionally secure verifiable scheme is protected against more than $n/3$ colluders; if more protection is needed, the scheme must either make computational assumptions or allow for a small probability of error. By allowing for a small probability of error, the scheme proposed by Rabin and Ben-Or [19] makes no computational assumptions, and allows for both a dishonest dealer and dishonest participants. The verifiable proactive scheme constructed by Stinson and Wei [23] is unconditionally secure and protects against $n/4 - 1$ colluders. While this resistance is less than optimal, it both simplifies the scheme and prevents secret information from being shared when participants are dishonest.

2.3 Proactive Schemes

Consider a mobile adversary [18], who may be able to infiltrate and gather the shares of an increasing number of participants over time. How can we protect our scheme against such an attacker? One possible approach is to refresh the secret periodically, but such an approach may not be desirable if the secret is difficult or expensive to change. If one imagines a document being encrypted with a key, and that key being used as the secret, refreshing the secret would require that the document be decrypted and re-encrypted with the old key, exposing it for a brief period each time the secret was changed. To address this problem, the notion of proactive secret sharing schemes was introduced by Herzberg et al. [13].

However, their approach relies on the computational difficulty of the discrete logarithm problem. Given the rapid rate at which previous cryptographic schemes have become obsolete because of technological advances, significant effort has been expended on the development of schemes that do not depend on any computational assumptions. Given that *proactive secret sharing schemes* are aimed to protect secrets that are difficult or expensive to change, it would seem especially beneficial in this scenario to choose a scheme whose security is unconditional.

2.4 Weighted Schemes

Broadly speaking, *weighted threshold secret sharing schemes* are designed so that participants deemed more powerful can recover the secret with the aid of fewer other participants. Instead of requiring t participants to recover the secret in a (t, n)-secret sharing scheme, we assign each participant a weight w_i and set up the scheme such that a set of participants A can recover the secret if and only if $\sum_{P_i \in A} w_{P_i} \geq t$. For example, it may be policy that the president and chief executive of the company have the collective authority to open the company safe, but that any two vice-presidents can substitute for a missing party in their absence.

A weighted scheme can be trivially constructed from any threshold scheme by assigning more shares to more powerful participants [3]. However, much of the work on weighted secret sharing schemes has focused on improving the ratio of the secret size to maximum share size, also known as the *information rate* [16].

2.5 Dynamic Schemes

The goal is to modify the access structure after the initialization. In fact in *dynamic secret sharing schemes*, there exist different access structures and the scheme is able to activate various of them at subsequent times [6]. If the dealer remains active for the entire duration of the scheme and has uninterrupted access to the participants through private channels, the access structure can be modified by simply creating shares for a new scheme with the desired access structure. Two more restricted variants of this approach have been studied: one in which the dealer is still functional, but only has access to a public broadcast channel, and one in which the dealer is no longer available after the scheme is initialized, but a private channel exists between each pair of participants.

In the former case, schemes have been proposed for single changes to threshold access structures, including the removal of a participant, known as disenrollment [4], or the modification of the threshold [1]. In the latter situation more progress has been made. One trivial solution has the participants each transfer the shares of their personal scheme to the corresponding shareholders of the new scheme [14]. Any authorized set of participants will be able to recombine all of the participants' shares, and thus recover the secret. However, such an approach results in each participant storing significantly more share data with each change in access structure; if repeated changes were made, both the storage requirements and recombination times would be unreasonable [15].

3 Our Construction

3.1 Setting

The proposed model consists of n participants, $P_1, P_2, \ldots, P_n$, and a dealer. The dealer will be present only during the initial set-up of the scheme. We assume the existence of private channels between each pair of participants, and that the dealer can communicate privately with participants in the dealing stage. We also consider the existence of an authenticated public broadcast channel, on which information is transmitted instantly and accurately to all participants. Each server must have a source of randomness. Finally, we need a synchronized source of time, to enable the periodic share update process.

In the model, the dealer and thus the information used to initialize the scheme, is no longer present after the initialization. Thus, all information used in the weight tuning process will be recovered from the participants of the scheme. Our objective is to protect the scheme against b colluders, we assume that the refreshing phase is instantaneous, and thus that the adversary cannot corrupt additional participants while shares are being updated and the weight tuning is taking place. A corrupted participant may send incorrect data to other players, both through the public broadcast channel or through the private channels.

3.2 Social Secret Sharing Scheme

In this construction, participants are assigned weights based on their behaviors in the model. Consequently, each participant receives a number of shares corresponding to its trust value which is the representation of a player's reliability over time. During the share refresh phase, the weights of players are adjusted in a way that players who cooperate receive more shares than those who defect. In addition, the scheme is able to disenroll corrupted players entirely or enroll newcomers to the system. To recover the secret, total weight of authorized players must be equal or bigger than the threshold. On the other hand, the weight of each participant must be less than the threshold. i.e., $\sum_{P_i \in A} w_{P_i} \geq t$ and $w_{P_i} < t$ for $1 \leq i \leq n$.

We consider a $n \times m$ matrix for the participants' identifiers, where n is the maximum number of participants and m is the maximum weight of each participant. As an example presented in Figure 1, assume we have four participants with different weights. After some period of time, suppose we observe defection (e.g., not being available to send S_4) from the first participant and cooperation from the fourth one. In that case, the system will decrease w_1 to 3 and increase w_4 to 2, that is the inactivation of the $id = 4$ and activation of the $id = 14$. The weight adjustment procedure is discussed in detail in Section 4.

In the next three discussions, we review the proposed constructions in [23, 9] and modify those protocols accordingly in order to fit them to our social secret sharing scheme with dynamic weighting. Let $GF(q)$ be a finite field and let ω be a primitive element in this field; all the computations are performed in the field $GF(q)$. In the descriptions below, a particular participant's identifiers are denoted by $m.ids$: meaning my ids, and the identifiers of other participants are denoted by $c.ids$: meaning colleagues' ids.

Verifiable Initialization. The presented protocol for initialization is an unconditionally secure verifiable secret sharing with $b \leq \frac{n}{4} - 1$ and $t > b + 1$. In our construction b is the total weight of colluders. This scheme will be used in the share renewal phase afterward. It also has a property that any coalition of $t - 1$ participants cannot get any information about the secret. Moreover, the dealer is not required to participate in the scheme after the initial distribution of secret information.

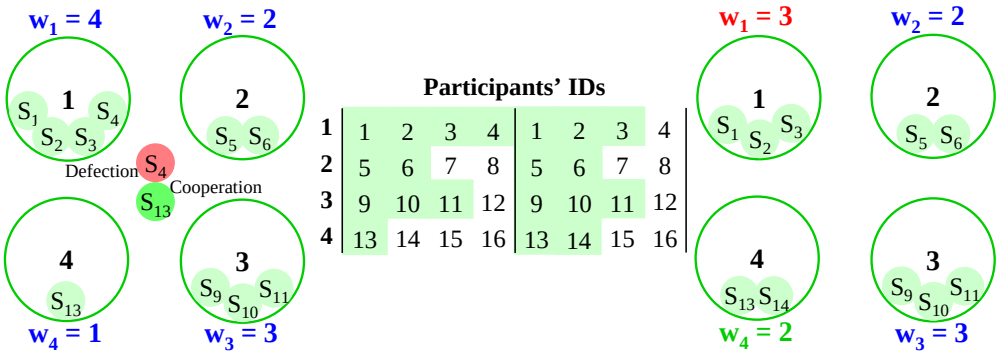

Fig. 1. Social secret sharing scheme

1. Dealer chooses a random symmetric polynomial of degree $t-1$ to generate shares for participants.

$$f(x,y) = \sum_{i=0}^{t-1}\sum_{j=0}^{t-1} a_{ij}x^i y^j$$

where $a_{00} = Secret$ and $a_{ij} = a_{ji}$ for all i, j. Then, dealer sends $h_{id}(x) = f(x, \omega^{id})$ to P_{id} through a private channel. For instance, if the first participant's weight is 2, then it has two activated *ids* and consequently receives two shares from the dealer.
2. In order to verify distributed shares, each player sends the value of $h_{m.ids}(\omega^{c.ids})$ to other players and receives the value of $h_{c.ids}(\omega^{m.ids})$ from them, all through secure channels, where all $ids \in \{activated.ids\}$.
3. Then, each participant checks to see whether $h_{m.ids}(\omega^{c.ids}) = h_{c.ids}(\omega^{m.ids})$. If a participant finds that they are not equal, then he broadcasts $(m.ids, c.ids)$.
4. Each P_{id} computes the subset $G \subseteq \{1, ..., n\}$ such that any ordered pair $(ids, ids) \in G \times G$ is not broadcasted, i.e. subset of good players. If $|G| \geq n - b$, then P_{id} outputs $ver_{id} = 1$, i.e. by excluding bad participants the verification has been satisfied. Otherwise, P_{id} outputs $ver_{id} = 0$.

If at least $n - b$ players output $ver_{id} = 1$, then dealer erases all info on his end; otherwise, the dealer reboots the system for another initialization. As an example, consider the following parameters: total number of activated *ids* or weights $n = 4$, threshold $t = 3$, primitive element $\omega = 2$, prime number $q = 13$, and weights $w_1 = 2$ $(ids : 1, 2)$, $w_2 = 1$ $(id : 3)$, and $w_3 = 1$ $(id : 5)$. We randomly generate the following bivariate symmetric polynomial with the constant term $a_{00} = 11$ as the secret value, and replace ω^{ids} with y to generate shares for activated *ids*: $f(x,y) = 11 + 3x + 3y + 2x^2 + 2y^2 + xy^2 + x^2y + 2xy + x^2y^2$, e.g., $f(x, 2^5) = 10 + 12x + 5x^2$: first share of the third participant. As stated earlier, each participant then plugs in $\omega^{c.ids}$ into x in order to send some values to his colleagues in the verification step, Figure 2.

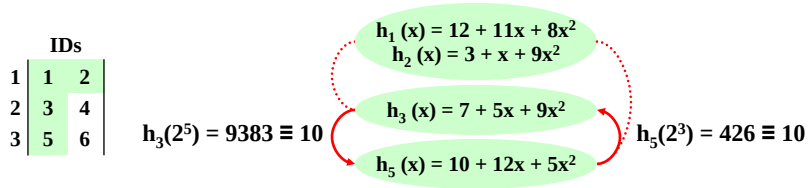

Fig. 2. Verifiable initialization

Verifiable and Proactive Share Renewal. After the initial step, we need to update participants' shares periodically in order to reduce the adversary power and change the number of shares each participant receives based on its new trust value. As stated in [9], the following procedure is used.

1. Each P_{id} selects a random symmetric polynomial of degree $t-2$ to generate new shares for himself and other participants, Figure 3.

$$r^{id}(x,y) = \sum_{i=0}^{t-2}\sum_{j=0}^{t-2} r_{ij}x^i y^j$$

where $a_{00} = 0$ and $r_{ij} = r_{ji}$ for all i, j.
2. P_{id} sends $r^{id}(x, \omega^k)$ to P_k through a private channel, where $k \in \{activated.ids\}$, Figure 3.

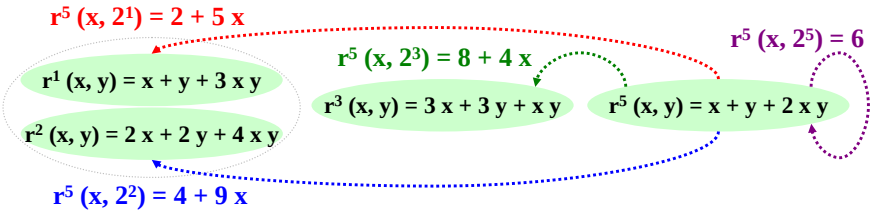

Fig. 3. New share generations

When the share distribution procedure is completed, Figure 4, each P_{id} has an original share of degree $t-1$ and as many new shares as activated ids with degree of at most $t-2$.

r1 (x, 21) = 2 + 7 x	r1 (x, 22) = 4	r1 (x, 23) = 8 + 12 x	r1 (x, 25) = 6 + 6 x
r2 (x, 21) = 4 + 10 x	r2 (x, 22) = 8 + 5 x	r2 (x, 23) = 3 + 8 x	r2 (x, 25) = 12
r3 (x, 21) = 6 + 5 x	r3 (x, 22) = 12 + 7 x	r3 (x, 23) = 11 + 11 x	r3 (x, 25) = 5 + 9 x
r5 (x, 21) = 2 + 5 x	r5 (x, 22) = 4 + 9 x	r5 (x, 23) = 8 + 4 x	r5 (x, 25) = 6
h1 (x) = 12 + 11x + 8x2 , h2 (x) = 3 + x + 9x2		h3 (x) = 7 + 5x + 9x2	h5 (x) = 10 + 12x + 5x2

Fig. 4. New share distributions is completed

3. In order to verify new distributed shares, each player sends the value of $r^l(\omega^{c.ids}, \omega^{m.ids})$ to other players and receives the value of $r^l(\omega^{m.ids}, \omega^{c.ids})$ from them, all through secure channels, where $l \in \{activated.ids\}$, Figure 5.

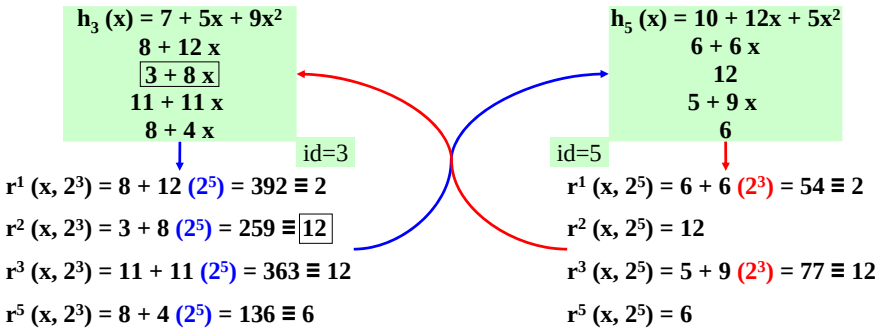

Fig. 5. Verification procedure

4. After that, each participant checks to see whether $r^l(\omega^{c.ids}, \omega^{m.ids}) = r^l(\omega^{m.ids}, \omega^{c.ids})$. If the equation is not true for *more than* b times (i.e., excluding colluders), then he broadcasts an accusation of P_l.

For example, suppose values that $P_{id=3}$ is generating with his *second* share: $r^2(x, 2^3) = 3 + 8x$, does not match with its corresponding values that he has received from other players: $r^2(x, 2^1) = 4 + 10x$, $r^2(x, 2^2) = 8 + 5x$, and $r^2(x, 2^5) = 12$ for more than b times, then $P_{id=3}$ broadcasts an accusation of $P_{id=2}$, Figure 6.

5. if P_{id} is accused by *at most* b players, the accusations could be from colluders. In this case, the participant can defend himself by broadcasting all the shares that he gave to those accusers $r^{id}(x, \omega^{ac})$, where $ac \in \{accusers' ids\}$. As an example showed in Figure 6, suppose $P_{id=3}$ broadcasts an accusation of $P_{id=2}$. In that case, $P_{id=2}$ can broadcast his second share, $3 + 8x$, to other players, so that they can judge who is being honest. Then, other players whose *ids* do not belong to the conflicting parties, check those values and broadcast *Yes* or *No*. If there are at least $n - b - c$ players broadcasting *Yes*, then $P_{id=2}$ is not a bad participant, where c is the number of activated *ids* belonging to the conflicting parties, e.g., $P_{id=1}$ and $P_{id=2}$ cannot judge since they belong to the first participant.

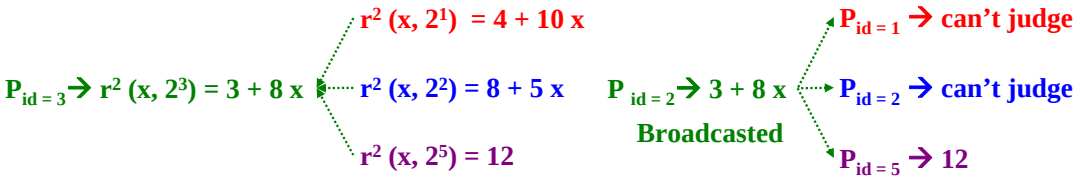

Fig. 6. Accusation and defense

6. Finally, each participant updates the list of good players G and his share as follows:

$$h_{id}(x) \longleftarrow h_{id}(x) + (x + \omega^{id}) \sum_{g \in G} r^g(x, \omega^{id})$$

For instance, if $\{1, 2, 3, 5\} \in G$, then the updated share of the $P_{id=3}$ would be as follows: $h_3(x) = 7 + 5x + 9x^2 + (x + 2^3)[(8 + 12x) + (3 + 8x) + (11 + 11x) + (8 + 4x)] = 5x^2 + 3x$. As you can see, the new share is a polynomial in x of degree $t - 1$.

Secret Recovery. To recover the secret either after the initialization or at any time later on, the following steps must be done:

1. Each P_{id} send $h_{id}(0)$ to P_k, where $id \in G$: good players.
2. Then, P_k computes a polynomial $f_k(0, \omega^{id}) = h_{id}(0)$ for at least $n - 2b$ of the data he received.
3. Finally, P_k computes the secret $= f_k(0, 0)$.

Now, we present a new construction for enrolling new players to a threshold secret sharing scheme.

Activating Players' Identifiers. To update shares in a proactive scheme, a participant must have its previous share. Suppose we intend to activate a new id in period p while we do not have its corresponding share in period $p - 1$. This problem can be resolved only if t participants cooperate together in order to generate the old share for the newcomer, where t is the threshold. For sake of simplicity, suppose each participant has only one identifier.

1. First, each participant P^i computes its corresponding Lagrange interpolation constant:

$$C_i = \prod_{1 \le j \le t, j \ne i} \frac{x - x_j}{x_i - x_j}$$

where x is the newcomer's id, x_i is the id of P^i, and x_j represents other participants' *ids*.

2. Second, participants multiply their shares in related Lagrange interpolation constants. Afterward, each participant randomly splits the result into t portions, i.e., $S_1 \times C_1 = s_{1,1} + s_{2,1} + \cdots + s_{t,1}$, $\cdots$, $S_t \times C_t = s_{1,t} + s_{2,t} + \cdots + s_{t,t}$.
3. Third, each participant keeps one share-portion for himself, and exchanges the rest of them with other $t-1$ players. As a result, each player P^j holds t values; so, he adds those values together and sends the result to the new participant: $v_j = \sum_{i=1}^{t} s_{j,i}$, where $s_{j,i}$ is the j^{th} share-portion of the i^{th} participant.
4. Finally, the new participant adds all these values together in order to construct his share: $S = \sum_{j=1}^{t} v_j$.

For instance, assume $t = 3$ and we have generated required shares for three participants with $ids = 1, 2, 3$ based on $f(x) = 3 + 2x + x^2$ in period $p - 1$, $f(1) = 6$, $f(2) = 11$, and $f(3) = 18$. After some time, we are asked to create a share for a new player, $id = 4$, without having access to the original polynomial. First each player P_i computes $S_i \times C_i$ as follows: $S_1 \times C_1 = 6 \times \frac{(4-2)(4-3)}{(1-2)(1-3)} = 6$, $S_2 \times C_2 = 11 \times \frac{(4-1)(4-3)}{(2-1)(2-3)} = -33$, and $S_3 \times C_3 = 18 \times \frac{(4-1)(4-2)}{(3-1)(3-2)} = 54$. Then, they cooperate to create a share for the newcomer, as shown in Figure 7.

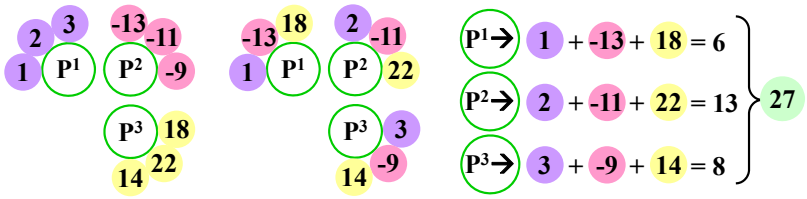

Fig. 7. New id activation

We can extend this approach to the situation where the share generator is a bivariate polynomial. Consider the bivariate Lagrange interpolation formula, where p is a prime, $x_1, x_2, ..., x_t$ are distinct elements in $\mathbb{Z}_p$, and $a_1(x), a_2(x), ..., a_t(x) \in \mathbb{Z}_p[x]$ are polynomial of degree at most $t-1$. Then there is a unique polynomial $r(x, y) \in \mathbb{Z}_p[x, y]$ having degree at most $t-1$ in x and y [22].

$$r(x, y) = \sum_{i=1}^{t} a_i(x) \prod_{1 \le j \le t, j \ne i} \frac{y - y_j}{y_i - y_j}$$

Suppose $t = 2$, $p = 13$, $\omega = 2$, activated $ids = \{1, 2\}$, and $f(x, y) = 11 + 2y + 2x + 3xy$ is the share generator. Therefore, the subsequent shares for the first and second ids are $f(x, 2^1) = 2 + 8x$ and $f(x, 2^2) = 6 + x$, which can be divided into pieces for distribution. If we intend to activate $id = 6$, then the corresponding share is as follows:

$$r(x, 2^6) = [(2 + 8x) \times \frac{2^6 - 2^2}{2^1 - 2^2}] + [(6 + x)\frac{2^6 - 2^1}{2^2 - 2^1}] = 9 + 12x$$

In the case of using symmetric bivariate polynomial for the share creation, the new player is able to verify the correctness of his share by checking $h_{m.ids}(\omega^{c.ids}) = h_{c.ids}(\omega^{m.ids})$ values with all other players. As an example mentioned earlier, the first player $P_{id=1}$ sends $2 + 8(2^6) \equiv 7$ to the newcomer $P_{id=6}$, and then he verifies that with $9 + 12(2^1) \equiv 7$.

4 Weight Adjustment

Our scheme provides a mechanism for assigning new weights to participants based on their behaviors at the end of each time period, by behavior we refer to a participant's credibility. We intend to apply a weight adjustment technique that supports reliable participants because of their repeated cooperation, reduces the influence of unreliable players due to their past defection, and protects the scheme from malicious players or active adversary. Since players' weights and trust values are public information, therefore, the trust computation and weight adjustment can be done by any authority or a committee of players.

4.1 Computing Trust Function

We first define a *reliability function*, $R^i(p) : \mathbb{N} \rightarrow \{0, 1\}$ for a participant P^i. In the following arguments, we use superscript i in order to refer to the i^{th} participant not players' *ids*. $R^i(p) = 1$ if P^i is cooperating in period p, otherwise $R^i(p) = 0$. To define how participants' trust values change over time, we need a function that determines an appropriate trust value given past behaviors, we call this the *trust function*.

Definition: *Given a participant P^i and a reliability function R^i for that, a function $T^i : \mathbb{N} \rightarrow \mathbb{R}$ is called a trust function for R^i, if it satisfies the following properties:*

1. $-1 \leq T^i(p) \leq +1 \ \forall p \in \mathbb{N}$
2. $T^i(p) \geq T^i(p-1)$ *whenever* $R^i(p) = 1$
3. $T^i(p) \leq T^i(p-1)$ *whenever* $R^i(p) = 0$

In fact, the trust function illustrates how reputable or trustworthy each participant is. We then let $T^i(p)$ represents user-i's trust value during period p. The definition for $T^i(p)$ must depend on $T^i(p-1)$. The function's recursive definition also requires that we set initial trust values $T^i(0)$ for participants; a solution would be to give each player the same initial value. One simple method is to assign an initial value to newcomers, increase the trust value by a constant factor if the participant is cooperating, and decrease it otherwise. However, this approach does not consider various scenarios when making the adjustment.

A new approach for trust management demonstrated by Nojoumian and Lethbridge [17], which supports good players strongly, blocks bad ones, and creates opportunities for newcomers or players who want to show their merit in a social network. They considered the following six scenarios in order to compute a trust function $T(x)$ based on players' historical behaviors, where $-1 \leq T(x) \leq +1$ and α, β define boundaries on trust values for bad, good, and new players.

1. If a bad player cooperates, then encourage him a little bit by the *encouragement* factor X_{Enc}, e.g., $X_{Enc} \in (0.01, 0.05)$.
2. If the scheme encounters a player who is looking for a chance by cooperating, then *give* him an opportunity by the factor X_{Give}, e.g., $X_{Give} = 0.05$.
3. If a good player cooperates, then reward him more than the encouragement factor by the *reward* factor X_{Rew}, e.g., $X_{Rew} \in (0.05, 0.09) > X_{Enc} \in (0.01, 0.05)$.
4. If a good player defects, then discourage him a little bit by the *discouragement* factor X_{Dis}, e.g., $X_{Dis} \in (-0.05, -0.01)$.
5. If the scheme encounters a player who cannot be judged while he is defecting, then *take* an opportunity from him by the factor X_{Take}, e.g., $X_{Take} = -0.05$.
6. If a bad player defects, then penalize him more than the discouragement factor by the *penalize* factor X_{Pen}, e.g., $|X_{Pen}| \in |(-0.09, -0.05)| > |X_{Dis}| \in |(-0.05, -0.01)|$.

By assigning different values to the above parameters, we can define various points and construct an appropriate trust function via regression. It is also desirable to bound the trust value, both above and below, so that a participant cannot continually build up reputation in order to be the main shareholder and form a monopoly; this also protects the scheme in a scenario where a malicious player cooperates for a while in order to gather most of the shares for a severe damage. Moreover, since the scheme detects corruptions and disenrolls bad players immediately, there is not any chance for an active adversary to continuously harm the scheme by sending wrong shares.

Furthermore, in scenarios where cooperation or defection is a decision on the part of the participant, such a transition function is still reliable. Consider the scenario in which a player cooperates on secret recoveries in regular situations (cheap cooperations) until reaching a high trust value, at which point he may defect the secret recovery in a sensitive situation (an expensive defection) without significant effect on his reputation value. The authors in [17] define the parameter λ as the *transaction cost*. In that case, the scheme would be able to fairly deal with the players' cooperation and defection.

4.2 Inactivating Unreliable Players' Identifiers or Disenrolling Bad Participants

Now that we have a trust value for each participant, we turn to the task of using that value to adjust the scheme. Clearly, *ids* should be inactivated for players whose trust values have been dropped and activated for those whose trust values are risen or for newcomers. The task is to determine how many *ids* should be inactivated for unreliable participants, and how other *ids* should be activated for cooperative players and new participants.

One option for ids' inactivation is simply to turn an *id* off for a player each time his trust value decreases. However, such an approach does not take into account the number of shares in the system, nor does it consider the number of shares the participant has: if the participant has a large number of shares, inactivation of a single *id* will have a negligible effect. Moreover, a participant with only one share remaining would effectively be removed from the system. One particular approach is to inactivate a number of *ids* proportional to the amount that the users' trustworthy dropped:

$$w^i(p) = \left\lfloor w^i(p-1) \cdot (1 - \frac{\gamma}{2}) \right\rfloor \text{ if } R^i(p) = 0$$

where $\gamma = T^i(p-1) - T^i(p)$ is the coefficient of the weight reduction. If $w^i(p)$ becomes zero, then the participant P^i will be removed from the scheme, i.e. the release of a row in the matrix of participants' identifiers. Consequently, the total number of *ids* to be activated is given as follows:

$$\delta(p) = \sum_{\{i\,:\,R^i(p)=0\}} [w^i(p-1) - w^i(p)]$$

As an example, suppose that trust values of a bad player P^i and a good player P^j have been dropped from $T^i(p-1) = -0.2$ to $T^i(p) = -0.8$ and $T^j(p-1) = 0.8$ to $T^j(p) = 0.2$ accordingly. In that case, the weight reduction's coefficient $\gamma = 0.6$ would be the same due to the symmetric range of the trust function, i.e., P^j has done an expensive defection compared to the P^i so that they got the same trust value reduction. In other words, we deduct the weight of unreliable and reliable players with the same coefficient since we have already considered their historical behaviors in the trust adjustment phase. Since $trust.value \in [-1, +1]$, we divide γ by 2 in order to compute the rate of the weight reduction in the [0,1] interval.

4.3 Activating Reliable Players' Identifiers or Enrolling New Participants

Given the number of *ids* to be activated, we now define which players should receive extra shares and how many newcomers can enter into the scheme. For each participant P^i, consider the ratio of a player's trust value $T^i(p)$ to the number of shares he is holding $w^i(p)$. This ratio increases with the participant's trust value enhancement, and decreases as the participant gains more shares.

As a result, it would be reasonable to activate *ids* in participants for whom this ratio is highest, but it is not enough since we also need to consider newcomers whose trust values are equal to zero. Therefore, in order to have a fair policy, we give the first priority to cooperative participants for whom the mentioned ratio is both highest and positive, the second priority to newcomers, and the third priority to other cooperative players, i.e., players with negative trust values. However, the re-distribution procedure must ensure that no player ends up with t or more shares. All these considerations suggest a technical solution, Algorithm 1, to determine new weights for most deserving participants and to add new players with a trust value of zero and a weight equal to one, where there are $\delta(p)$ *ids* to be activated.

The proposed algorithm assumes that there will be enough cooperative players and newcomers to activate their *ids*, who do not already have $t-1$ shares. Given that our underlying scheme handles at most b colluders in a given time period, the scenario in which there are no cooperative participants or newcomers to receive shares seems unlikely. However, our algorithm can easily be modified to handle this situation by assigning the remaining shares to participants who, despite having behaved irresponsibly in the previous time period, still have relatively high trust values. Doing so maintains a constant number of shares in the

Algorithm 1 Activating identifiers for cooperative players and newcomers

for $i = 1$ to $k = \delta(p)$ **do**
  select P^j {with one the following conditions}
  if $R^j(p) = 1$ and $w^j(p) < t-1$ and $T^j(p) > 0$ and $\frac{T^j(p)}{w^j(p)}$ is highest **then**
    activate a new id for P^j
    $w^j(p) = w^j(p-1) + 1$
  else if $P^j \in \{newcomers\}$ **then**
    assign a new row to P^j
    activate a new id for P^j
    $w^j(p) = 1$
    $T^j(p) = 0$
  else if $R^j(p) = 1$ and $w^j(p) < t-1$ and $T^j(p) < 0$ and $\frac{T^j(p)}{w^j(p)}$ is highest **then**
    activate a new id for P^j
    $w^j(p) = w^j(p-1) + 1$
  end if
end for

system. As given, the complexity of the algorithm is $O(kn)$ where $k = \delta(p)$ and n is required to locate the highest $T^j(p)/w^j(p)$, but it could be modified to run in $O(k + n \log n)$ time by sorting the participants' trust-to-shares ratios.

To add participants to the scheme, we have two options for assigning *ids* to new players in the identification matrix. The first solution is to add a row for each new player in the matrix. As time passes, this approach leads to a big matrix with empty rows and consequently increases the size of identifiers. The second alternative is to use released rows of former bad players. Since we first remove bad players from the scheme and then remaining participants update their shares; therefore, the leaving players just hold a number of old shares less than $t-1$, that is because of our initial assumption $b < t-1$. As a consequent, we can reuse the released *ids* of bad players and assign them to newcomers without leaking any information about the secret because new players will receive updated shares corresponding to those recycled *ids*.

5 Security Analysis

The security of the proposed protocols can be illustrated by the fact that the secret remains the same and the updated shares in the consequent steps cannot be computed by colluders.

Regarding the first property, the secret is the constant term of the original bivariate symmetric polynomial at the initialization step, let say f_1. Subsequently, each participant P_{id} generates a new random symmetric polynomial of degree $t-2$ with a constant term equal to zero, i.e., $r^{id}(x, y)$. At the end, when participants add their original shares, $h_{id}(x)$, to the new shares they have received from other players, $r^j(x, \omega^{id})$, the constant term of the original polynomial stays the same. As a result, the secret will be encoded in a new bivariate symmetric polynomial, let say f_2. Therefore, shares exposed by security breaches do not reveal any information about the secret unless an adversary collects t shares belonging to the same edition of the symmetric bivariate polynomial.

Concerning the second property, a group of colluders with total weight of b can compute at most $b < t-1$ points which give no information about the secret. In addition, because of the random polynomial generation at each share renewal phase, the adversary cannot use the information he gained in period p during period $p+1$ or in any other period.

There are two other concerns regarding the defense protocol and ids' activation procedure. First, a good player P_{id} broadcasts at most b polynomials in order to defend himself. Since $t > b+1$; therefore, the broadcasted polynomials, $r^{id}(x, \omega^{ac})$ where $ac \in \{accusers' ids\}$, give no information about $r^{id}(x, y)$. Second, when t participants want to activate a new id, each player randomly splits his share, keeps a fraction for himself, and exchanges $t-1$ portions with others. Even if $t-1$ players collude, the proposed procedure does not reveal a participant's share and consequently the secret, we prove this issue with the following theorem.

Theorem: *The proposed protocol for activating players' identifiers is an unconditionally secure scheme.*

Proof: Suppose each participant P^i first multiplies his share in corresponding Lagrange interpolation constant C_i. For the sake of simplicity, we consider this multiplication's result as the share of participant P^i. Since, each player splits his share into t pieces, we define a $n \times m$ matrix, where each column shows various fractions of a single share and each row represents portions of different shares that each player receives from other participants. In the other word, all values in j^{th} column, i.e., $s_{1,j}, s_{2,j}, \cdots, s_{t,j}$, belongs to a single player P^j and all entries in i^{th} row, i.e., $s_{i,1}, s_{i,2}, \cdots, s_{i,t}$, represent values that player P^i receives from other participants. We consider the following two scenarios to see if a coalition of $t-1$ participants can reveal any information regarding the secret.

Case-I: $t-1$ of t cooperating participants collude. In this case, colluders have access to all entries of $t-1$ rows. In addition, they also know $t-1$ entries of the unknown row because $t-1$ columns belong to them. Therefore, just one entry remains unknown which prevents colluders to find the newcomer's share and consequently the secret. As presented in Figure 7, if P^1 and P^2 collude, 14 in the third row remains unknown.

Case-II: $t-2$ of t cooperating participants plus the newcomer collude. In this case, colluders have access to all entries of $t-2$ rows, in addition, they also know $t-2$ entries of two unknown rows because $t-2$ columns belong to them; thus, four entries remain unknown. On the other hand, the newcomer also knows the summation of rows' entries for all rows, therefore, he can just construct two equations with four unknowns which does not reveal any information about the secret. As presented in Figure 7, if P^1 and the newcomer P^4 collude, the values -11 and 22 in the second row and the values -9 and 14 in the third row remain unknown and P^4 can only construct the following two equations: $2 + x_1 + x_2 = 13$ and $3 + x_3 + x_4 = 8$.

6 Conclusions and Future Work

The scheme that we have presented is an unconditionally secure, proactive secret sharing scheme with a dynamic access structure. Since a players's weight and consequently his influence on the entire scheme is dependent on the historical behaviors and participants interaction, we have termed our construction a *social secret sharing scheme*, where shares are allocated based on the trust value of each player and the way he interacts with other participants.

The proposed construction has a variety of desirable properties: it is *unconditionally secure*, meaning that it does not rely on any computational assumptions; *verifiable*, detecting defection of irresponsible players and cooperation of reliable players; *proactive*, refreshing shares at each cycle without changing the secret; *dynamic*, allowing changes to the access structure after the initialization; and *weighted*, allowing trusted participants to have more authority in the scheme. The other prominent property of the scheme is that, it gradually reduces the influence of unreliable players due to the *self-reinforcement* property of social interactions among players. In other words, players collaborate with those whom they really trust; conversely, they tend not to cooperate with those whom they do not trust. This issue creates an increasing gap between reliable and unreliable players unless a participant undergoes a sustained change in behavior. Applications of such a paradigm are: electronic auctions with private bids running by intelligent agents, joint signature, and shared decryption keys.

Future work could focus on extending the proposed scheme to other constructions. The initial scheme is based on bivariate symmetric polynomials, therefore, we would like to apply other mathematical models to form new constructions. The paradigm will be then extended to threshold changeable and non-threshold schemes, and other access structures such as hierarchy frameworks, where players are assigned to different levels in a hierarchy. In such schemes, a set of participants is authorized to recover the secret only if they are at sufficiently high levels of authority [25]. It would be also interesting to consider a scheme where the weight adjustment was done in a manner that allowed each participant to form an opinion of others, with each participant's opinions affecting the redistribution process. As a result, players could use a *referral chain*, allowing them to exchange their opinions or past experiences with respect to a specific player.

References

[1] Barwick, S. G., Jackson, W. A., and Martin, K. M. Updating the parameters of a threshold scheme by minimal broadcast. *IEEE Transactions on Information Theory 51*, 2 (2005), 620–633.

[2] Ben-Or, M., Goldwasser, S., and Wigderson, A. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC* (1988), ACM, pp. 1–10.

[3] Benaloh, J. C., and Leichter, J. Generalized secret sharing and monotone functions. In *CRYPTO* (1988), S. Goldwasser, Ed., vol. 403 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 27–35.

[4] Blakley, B., Blakley, G. R., Chan, A. H., and Massey, J. L. Threshold schemes with disenrollment. In *CRYPTO* (1992), E. F. Brickell, Ed., vol. 740 of *Lecture Notes in Computer Science*, Springer, pp. 540–548.

[5] Blakley, G. R. Safeguarding cryptographic keys. In *National Computer Conference, New York* (Montvale, NJ, USA, 1979), R. E. Merwin, J. T. Zanca, and M. Smith, Eds., vol. 48 of *AFIPS Conference proceedings*, AFIPS Press, pp. 313–317.

[6] Blundo, C., Cresti, A., Santis, A. D., and Vaccaro, U. Fully dynamic secret sharing schemes. *Theoretical Computer Science 165*, 2 (1996), 407–440.

[7] Boneh, D., and Franklin, M. Efficient generation of shared rsa keys. *Journal of ACM 48*, 4 (2001), 702–722.

[8] Chor, B., Goldwasser, S., Micali, S., and Awerbuch, B. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *FOCS* (1985), IEEE, pp. 383–395.

[9] D'Arco, P., and Stinson, D. R. On unconditionally secure robust distributed key distribution centers. In *Advances in Cryptology, Proceedings of ASIACRYPT '02, Lecture Notes in Computer Science* (2002), Springer-Verlag, pp. 346–363.

[10] Du, W., and Atallah, M. J. Secure multi-party computation problems and their applications: a review and open problems. In *NSPW '01: Proceedings of the workshop on new security paradigms* (2001), ACM, pp. 13–22.

[11] Goldwasser, S. Multi party computations: past and present. In *PODC '97: Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing* (1997), ACM, pp. 1–6.

[12] Harkavy, M., Tygar, J. D., and Kikuchi, H. Electronic auctions with private bids. In *WOEC'98: Proceedings of the 3rd Conference on USENIX Workshop on Electronic Commerce* (1998), USENIX Association, pp. 6–6.

[13] Herzberg, A., Jarecki, S., Krawczyk, H., and Yung, M. Proactive secret sharing or: How to cope with perpetual leakage. In *CRYPTO* (1995), D. Coppersmith, Ed., vol. 963 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 339–352.

[14] Martin, K. Dynamic access policies for unconditionally secure secret sharing schemes. In *ITW '05: Proceedings of IEEE Information Theory Workshop* (2005), pp. 61–66.

[15] Martin, K. M., Safavi-Naini, R., and Wang, H. Bounds and techniques for efficient redistribution of secret shares to new access structures. *The Computer Journal 42*, 8 (1999), 638–649.

[16] Morillo, P., Padró, C., Sáez, G., and Villar, J. L. Weighted threshold secret sharing schemes. *Information Processing Letters 70*, 5 (1999), 211–216.

[17] Nojoumian, M., and Lethbridge, T. C. A new approach for the trust calculation in social networks. In *E-business and Telecommunication Networks: 3rd International Conference on E-Business, Selected Papers* (2008), vol. 9 of *CCIS*, Springer-Verlag, pp. 64–77.

[18] Ostrovsky, R., and Yung, M. How to withstand mobile virus attacks (extended abstract). In *PODC '91: Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing* (1991), ACM, pp. 51–59.

[19] Rabin, T., and Ben-Or, M. Verifiable secret sharing and multiparty protocols with honest majority. In *STOC* (1989), ACM, pp. 73–85.

[20] Shamir, A. How to share a secret. *Communications of the ACM 22*, 11 (1979), 612–613.

[21] Stinson, D. R. An explication of secret sharing schemes. *Designs, Codes and Cryptography 2*, 4 (1992), 357–390.

[22] Stinson, D. R. *Cryptography: Theory and Practice,Third Edition.* CRC Press, 2005.

[23] Stinson, D. R., and Wei, R. Unconditionally secure proactive secret sharing scheme with combinatorial structures. In *Selected Areas in Cryptography* (1999), H. M. Heys and C. M. Adams, Eds., vol. 1758 of *Lecture Notes in Computer Science*, Springer, pp. 200–214.

[24] Tartary, C., and Wang, H. Dynamic threshold and cheater resistance for shamir secret sharing scheme. In *Inscrypt* (2006), H. Lipmaa, M. Yung, and D. Lin, Eds., vol. 4318 of *Lecture Notes in Computer Science*, Springer, pp. 103–117.

[25] Tassa, T. Hierarchical threshold secret sharing. *Journal of Cryptology 20*, 2 (2007), 237–264.

[26] Tompa, M., and Woll, H. How to share a secret with cheaters. *Journal of Cryptology 1*, 2 (1988), 133–138.

[27] Zhang, Y., and Liu, Z. Dynamic and verifiable secret sharing among weighted participants. *Journal of Systems Science and Complexity 20*, 4 (2007), 481–485.