# Novel Secret Sharing and Commitment Schemes for Cryptographic Applications

by

Mehrdad Nojoumian

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2012

## Abstract

In the second chapter, the notion of a *social secret sharing* (*SSS*) scheme is introduced in which shares are allocated based on a player's reputation and the way she interacts with other parties. In other words, this scheme renews shares at each cycle without changing the secret, and it allows the trusted parties to gain more authority. Our motivation is that, in real-world applications, components of a secure scheme have different levels of importance (i.e., the number of shares a player has) and reputation (i.e., cooperation with other parties). Therefore, a good construction should balance these two factors accordingly.

In the third chapter, a novel *socio-rational secret sharing* (*SRS*) scheme is introduced in which rational foresighted players have long-term interactions in a social context, i.e., players run secret sharing while founding and sustaining a public trust network. To motivate this, consider a repeated secret sharing game such as sealed-bid auctions. If we assume each party has a reputation value, we can then penalize (or reward) the players who are selfish (or unselfish) from game to game. This social reinforcement stimulates the players to be cooperative in the secret recovery phase. Unlike the existing protocols in the literature, the proposed solution is stable and it only has a single reconstruction round.

In the fourth chapter, a comprehensive analysis of the existing *dynamic secret sharing* (*DSS*) schemes is first provided. In a threshold scheme, the sensitivity of the secret and the number of players may fluctuate due to various reasons. Moreover, a common problem with almost all secret sharing schemes is that they are "one-time", meaning that the secret and shares are known to everyone after secret recovery. We therefore provide new techniques where the threshold and/or the secret can be changed multiple times to arbitrary values after the initialization. In addition, we introduce a new application of dynamic threshold schemes, named *sequential secret sharing* (*SQS*), in which several secrets with increasing thresholds are shared among the players who have different levels of authority.

In the fifth chapter, a cryptographic primitive, named *multicomponent commitment scheme* (*MCS*) is proposed where we have multiple committers and verifiers. This new scheme is used to construct different *sealed-bid auction protocols* (*SAP*) where the auction outcomes are defined without revealing the losing bids. The main reason for constructing secure auctions is the fact that the values of the losing bids can be exploited in future auctions and negotiations if they are not kept private. In our auctioneer-free protocols, bidders first commit to their bids before the auction starts. They then apply a decreasing price mechanism to define the winner and selling price in an unconditionally secure setting.

## Acknowledgements

I would like to take this opportunity to express my special gratitude to Douglas Stinson, for his endless support and excellent guidance. I have been greatly honored to work with him as an intelligent, enthusiastic, and responsible advisor. His patience, encouragement, and elegant thoughts have always brightened my path to the future and helped me to see the unseen barriers. I am forever thankful for all I have learned from him.

I highly appreciate Urs Hengartner and Ian Goldberg for their constructive and helpful comments that kept me on the right track. Urs has always prompted me to think about the applicability of my research projects. Ian has constantly helped me to have a deep understanding of complications behind mathematical constructions. Their suggestions led to a significant improvement of this thesis.

I also would like to extend my appreciation to Christos Papadimitriou, Charles Rackoff, Omer Reingold, and Ronald Cramer, who gave me opportunities to present different parts of this thesis at the University of California Berkeley, University of Toronto, Microsoft Research Silicon Valley, and Centrum Wiskunde and Informatica. Their helpful feedback improved this research extensively.

It is my pleasure to thank Reihaneh Safavi-Naini and Guang Gong, who accepted to read this thesis as the defense committee members. In addition, I greatly thank Keith Geddes, who gave me the opportunity to join the University of Waterloo, and Margaret Towell, who has significantly helped me with my scholarship and job applications. Special thanks to Timothy Lethbridge, my Master's thesis advisor, who has taught me a lot.

Many thanks to David Cheriton as well as Natural Sciences and Engineering Research Council of Canada (for NSERC CGS), Ministry of Training, Colleges and Universities, Canada (for OGS), and University of Waterloo (for Entrance and President's Scholarships) for proving funding to my Research. Finally, I would like to thank Atefeh, Jeremy, Aniket, Gregory, Ryan, Kevin, Jalaj, Colleen, and the other colleagues in the CrySP lab.

I would also like to extend my deepest gratitude to my family. My dear wife, Sareh Taebi, for her sincere love, positive energy, and endless support. Her smile has always given me hope and the greatest pleasure. My parents, Mehdi Nojoumian and Fatemeh Amin, for their unconditional love and support throughout my life. My elder sister and brother, Mahboubeh and Peyman, for their continuous encouragement. I have been fortunate to have such a great family. They all have had a constructive impact on my career.

**Dedication**

*To my ever supportive and loving wife, Sareh.*

# Table of Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | |
|---|---|
| AUF | Actual Utility Function |
| DSS | Dynamic Secret Sharing |
| LUF | Long-Term Utility Function |
| MCS | Multicomponent Commitment Scheme |
| MPC | Multiparty Computation |
| NE | Nash Equilibrium |
| PSS | Proactive Secret Sharing |
| RFP | Rational Foresighted Player |
| RSG | Repeated Social Game |
| RSS | Rational Secret Sharing |
| SAP | Sealed-Bid Auction Protocol |
| SNE | Social Nash Equilibrium |
| SQS | Sequential Secret Sharing |
| SRS | Socio-Rational Secret Sharing |
| SSS | Social Secret Sharing |
| TSS | Threshold Secret Sharing |
| VSS | Verifiable Secret Sharing |
| WSS | Weighted Secret Sharing |

# Notations

| | | | |
|---|---|---|---|
| $m$ | Maximum Weight | $D$ | Dealer |
| $n$ | Number of Players | $T$ | Trusted Initializer |
| $p$ | Period | $\mathcal{P}$ | Set of Players |
| $q$ | Prime Number | $P_i$ | Players |
| $t$ | Threshold | $B_i$ | Bidders |
| $\omega$ | Primitive Element | $\Upsilon$ | Access Structure |
| $a_i$ | Action | $\Delta$ | Uncorrupted Players |
| $\beta_i$ | Sealed-Bids | $\nabla$ | Corrupted Players |
| $\gamma_i$ | Lagrange Constants | $\mathcal{B}$ | Bad players |
| $u_i$ | Long-Term Utility Function | $\mathcal{N}$ | Newcomers |
| $u_i'$ | Actual Utility Function | $\mathcal{G}$ | Good Players |
| $w_i$ | Weight | $\mathcal{C}$ | Cooperation |
| $\vec{a}$ | Game's Outcome | $\mathcal{D}$ | Defection |
| $\vec{\sigma}$ | Players' Strategies | $\mathcal{X}$ | Corruption |
| $\mathcal{T}_i$ | Reputation | $\perp$ | No Participation |
| $\mathbb{Z}_q$ | Finite Field | $\mathcal{C}_{n\times n}$ | Pairwise Check Matrix |
| $\mathcal{V}_{n\times n}$ | Vandermonde Matrix | $\mathcal{E}_{n\times n}$ | Share Exchange Matrix |
| $\mathcal{Z}_{n\times n}$ | Projection Matrix | $\mathcal{M}_{n\times m}$ | Identifiers Matrix |

# Chapter 1

# Introduction

This thesis is focused on new cryptographic primitives and their applications in secure systems such as sealed-bid auctions. *Cryptographic primitives* are building blocks of secure systems. These primitives are designed by computer scientists and mathematicians in various models subject to certain conditions and assumptions. Scientists then provide proofs to show how a primitive complies with all the required specifications. Engineers later use these building blocks in real-world applications and software. We refer to secret sharing and commitment schemes as two major cryptographic primitives that are the main focus of this thesis.

In a *secret sharing scheme*, a dealer distributes partial information regarding a secret, called shares, among a set of players. Subsequently, players come together to reveal the secret in the absence of the dealer. In a *commitment scheme*, a player first commits to a secret without revealing it. He then discloses his secret to other parties. This is used to bind a player to a secret so that he cannot change it later. As an application of these primitives, we can refer to *secure auctions* where the bidders first submit their sealed-bids and later the auction outcomes (i.e., the selling price and winner) are defined without revealing the losing bids. Indeed, if the losing bids are not kept private, sellers can exploit them in the future auctions and negotiations in order to maximize their revenues, which would give them an unfair advantages.

In this chapter, an overview of the thesis is first provided, i.e., four new cryptographic primitives are introduced in different models with certain specifications. We also demonstrate how they can be used in various application scenarios. Subsequently, we illustrate some preliminaries to be used in further technical discussions.

## 1.1 Thesis Contributions

In the next two chapters, new secret sharing schemes in a social context as well as a game-theoretic model are introduced. In the next chapter, the existing dynamic secret sharing schemes are revisited and then new solutions are provided. In the final chapter, a multicomponent commitment scheme is proposed to be used in first-price sealed-bid auctions. We next provide a brief description with respect to the contributions of the thesis, also shown in Figure 1.1.

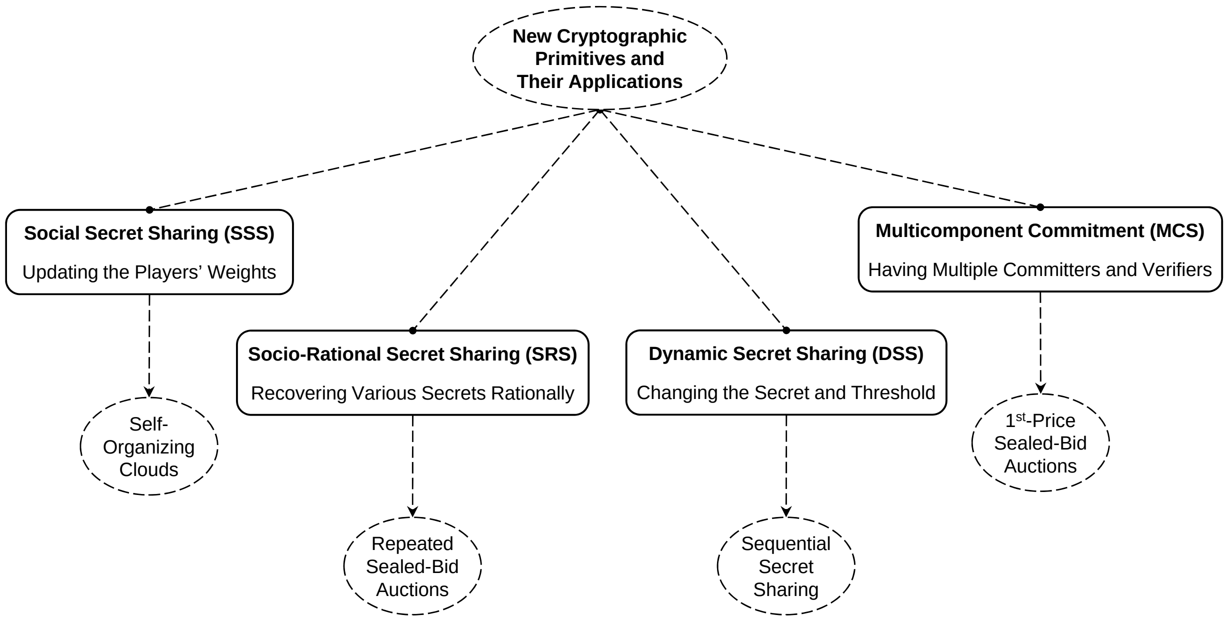
New Cryptographic Primitives and Their Applications
Social Secret Sharing (SSS)
Updating the Players' Weights
Socio-Rational Secret Sharing (SRS)
Recovering Various Secrets Rationally
Dynamic Secret Sharing (DSS)
Changing the Secret and Threshold
Multicomponent Commitment (MCS)
Having Multiple Committers and Verifiers
Self-Organizing Clouds
Repeated Sealed-Bid Auctions
Sequential Secret Sharing
1st-Price Sealed-Bid Auctions

Figure 1.1: Summary of the Thesis Contributions

We first introduce a new cryptographic primitive in a social context, named *social secret sharing* [76, 71]. In this scheme, shares of a secret are allocated based on each player's reputation portraying how cooperative she has been with other parties. In other words, the number of shares that each player receives at each cycle is updated according to her reputation; i.e., the trusted parties gain more shares compared to non-cooperative players. We motivate this construction with an application in *self-organizing clouds* [74], where the service providers who are available at the secret recovery phase receive more shares for the next cycle compared to the providers who are not available or respond with delay. This enables us to maintain a balance in the number of shares a player has and its availability for secret recovery.

Subsequently, we introduce a novel cryptographic primitive in a hybrid setting, named *socio-rational secret sharing* [75], that is, the game theoretic concept of rationality is consolidated by a social model. In this protocol, rational foresighted players run secret sharing while founding and sustaining a public trust network among themselves. We motivate this scheme with a *repeated sealed-bid auction* in which secret sharing is used to seal the proposed valuations. Since each player has a reputation value, we can then penalize (or reward) the players who are selfish (or unselfish) from game to game to incentivize them to be cooperative in the secret recovery phase. In other words, players who are cooperative have a greater chance to be invited to the future secret sharing games as opposed to non-cooperative players who act selfishly.

Next, we provide a comprehensive analysis of the previous *dynamic secret sharing* schemes [73]. In contrast to our previous chapters where we defined new models for two new cryptographic primitives, here we revisit the existing problem of threshold and secret changeability. In fact, in a threshold scheme, the sensitivity of the secret and the number of participants may change due to many reasons. In addition, secret changeability might be required from time to time. Therefore, we propose new methods to modify the threshold and/or the secret after the scheme's initialization. We motivate our dynamic schemes by a new application, named *sequential secret sharing*, where multiple secrets with increasing thresholds are shared among the players who have different levels of authority.

Finally, we propose a new cryptographic primitive, named a *multicomponent commitment scheme* [72], where we have multiple committers and verifiers. In this scheme, each member of a group of players commits to his secret value in the commitment phase. Subsequently, they collaborate to validate each secret in the reveal phase. We motivate this construction by presenting three different unconditionally secure *first-price sealed-bid auctions.* The major motivation for constructing secure auctions is that the losing valuations can be exploited in the future auctions and negotiations if they are not kept private. We should note that we apply a decreasing price mechanism without using any auctioneers in order to determine the winner and selling price; i.e., the bidders define the auction outcomes themselves.

## 1.2 Security Model

For further technical discussions, we first review various types of adversarial models in the setting of secret sharing. In our constructions, we consider two types of channels: *private channels*, which are secure and exist between each pair of players, and a *broadcast channel*, on which information is transmitted instantly and accurately to all parties.

- *Passive versus Active Adversary*: In the former case, the players follow protocols correctly but bad players may attempt to learn the secret. Such adversaries are also known as *honest-but-curious*. In the latter case, the players may deviate from protocols by sending incorrect shares (to prevent secret recovery or cause the reconstruction of an incorrect secret) while at the same time trying to learn the secret.

- *Static versus Mobile Adversary*: A passive or active adversary can be *static* or *mobile*. The former refers to an adversary who corrupts a fixed set of players ahead of time, while in the latter case, the adversary may corrupt different players at various stages of the protocol's execution.

- *Computational versus Unconditional Security*: In the former case, the security of the protocols relies on computational assumptions (e.g., the hardness of factoring or discrete logarithm), whereas in the latter case, the adversary is allowed to have unlimited computational power.

## 1.3 Secret Sharing

In a basic *secret sharing scheme*, we have a trusted dealer and $n$ players. The dealer choses the secret $\alpha$ from a specified set of possible secrets. Then the dealer creates partial information, called shares, and gives one share to each player. Later, a subset of players tries to reconstruct the secret from the shares they collectively hold. Before we give the security definition for secret sharing, we first provide the definition of an "access structure".

**Definition 1.1** *Let $\mathcal{P}$ be a finite set of $n$* players*. An* access structure $\Upsilon$ *is a set of subsets of players (called* authorized subsets) *that satisfies two conditions:*

*(a) if $A \in \Upsilon$ and $A \subseteq B \subseteq \mathcal{P}$, then $B \in \Upsilon$, and*

*(b) if $A \in \Upsilon$ then $|A| > 0$.*

*In a* threshold access structure*, the authorized sets are all sets of players $A$ such that $|A| \geq t$ where $t$ is the threshold of the scheme.*

A secret sharing scheme must satisfy the two following properties:

1. **Correctness**: if the players in an authorized subset combine their shares, then they can compute the secret.
2. **Secrecy**: on the other hand, if the players in an unauthorized subset combine their shares, then they have no information about the value of the secret.

In a $(t, n)$-*threshold secret sharing* (*TSS*) scheme [94, 11], any $t$ players can combine their shares to reveal the secret, but no set of $t-1$ players can learn any information about the secret. That is, the access structure consists of all set of $t$ or more players in $(t, n)$-threshold secret sharing.

### 1.3.1 Threshold Secret Sharing

In this section, we first recall the Lagrange interpolation method [97] and then we review a threshold secret sharing scheme.

Let $q$ be a prime number. Let $x_1, ..., x_t$ be distinct elements in the finite field $\mathbb{Z}_q$ and let $f_1, ..., f_t$ be arbitrary elements in $\mathbb{Z}_q$. Then there is a unique polynomial $f(x) \in \mathbb{Z}_q[x]$ of degree at most $t-1$ such that $f(x_i) = f_i$ for $1 \le i \le t$:

$$f(x) = \sum_{i=1}^{t} \Bigg( \prod_{1 \le j \le t, j \ne i} \frac{x - x_j}{x_i - x_j} \times f_i \Bigg). \tag{1.1}$$

Lagrange interpolation can also be applied to bivariate polynomials. Let $y_1, ..., y_t$ be distinct elements in $\mathbb{Z}_q$ and let $f_1(x), ..., f_t(x)$ be polynomials of degree at most $t-1$ in $\mathbb{Z}_q[x]$. Then there is a unique polynomial $f(x, y) \in \mathbb{Z}_q[x, y]$ of degree at most $t-1$ such that $f(x, y_i) = f_i(x)$ for $1 \le i \le t$:

$$f(x, y) = \sum_{i=1}^{t} \Bigg( \prod_{1 \le j \le t, j \ne i} \frac{y - y_j}{y_i - y_j} \times f_i(x) \Bigg). \tag{1.2}$$

When we say that a bivariate polynomial $f(x, y)$ has *degree at most* $t-1$, we mean that any term of the form $cx^i y^j$ in $f(x, y)$ has $i \le t-1$ and $j \le t-1$.

As a realization of threshold secret sharing, we can refer to a *Shamir threshold scheme*, which consists of the following two phases:

1. **Secret Sharing**: a dealer $D$ selects a random polynomial $f(x) \in \mathbb{Z}_q[x]$ of degree at most $t-1$ such that its constant term is the secret, i.e., $f(0) = \alpha$. He then sends the share $f(i)$ to $P_i$ for all $1 \leq i \leq n$, i.e., each player receives a point on the secret sharing polynomial.

2. **Secret Recovery**: subsequently, any subset $\Delta$ of at least $t$ players can collaborate to reconstruct the secret by Lagrange interpolation, in the absence of the dealer:

$$f(0) = \sum_{i \in \Delta} \Big( \prod_{j \in \Delta, j \neq i} \frac{j}{j-i} \times f(i) \Big). \tag{1.3}$$

Although any $t$ players can combine their shares to reveal the secret, no set of $t-1$ parties can learn the secret. For future use, let

$$\gamma_i^{\Delta} \overset{\text{def}}{=} \prod_{j \in \Delta, j \neq i} \frac{j}{j-i}$$

denote the *Lagrange constants* used in the above formula, where $j \in \Delta$ and $j \neq i$. Here we provide a toy example of the Shamir threshold scheme:

**Example 1.2** *The dealer selects a secret sharing polynomial* $f(x) = \boldsymbol{5} + 3x + 6x^2 \in \mathbb{Z}_{13}[x]$. *He then distributes the following shares among* $P_1, P_2, P_3, P_4$ *and leaves the scheme:*

$$f(1) = 1, f(2) = 9, f(3) = 3, f(4) = 9.$$

*Any three players, say* $P_1, P_2, P_3$*, can pool their shares to recover the secret by Lagrange interpolation as follows:*

$$\begin{aligned} f(0) &= \Big(\frac{2}{2-1}\Big)\Big(\frac{3}{3-1}\Big)(1) + \Big(\frac{1}{1-2}\Big)\Big(\frac{3}{3-2}\Big)(9) + \Big(\frac{1}{1-3}\Big)\Big(\frac{2}{2-3}\Big)(3) \mod 13 \\ &= -21 \mod 13 = \mathbf{5}. \end{aligned}$$

### 1.3.2 Other Types of Secret Sharing Schemes

In a *verifiable secret sharing scheme* (*VSS*) [24], the dealer and/or some of the players might be malicious. It is required that the players be able to verify the consistency of their shares after the scheme's initialization. If a sufficient number of players attest to the consistency of their shares, then they will be able to reconstruct a unique secret. In the next section, we review a simple verifiable secret sharing scheme and discuss its properties in detail. In a threshold VSS scheme, the number of malicious players must be less than the threshold $t$.

The papers [9, 23] provide the first unconditionally secure VSS when $t < \frac{n}{3}$ with a zero probability of error. They only assume the existence of secure private channels between each pair of participants. Later, [86, 8] utilize both private channels and a broadcast channel in order to construct a new VSS when $t < \frac{n}{2}$. This protocol has a negligible probability of error. To simplify previous constructions, [98, 35] propose VSS schemes based on symmetric bivariate polynomials in an unconditionally secure setting. These constructions assume secure private channels and a broadcast channel under the assumption that $t < \frac{n}{4}$.

The notion of *proactive secret sharing scheme* (*PSS*) is introduced in [44]. PSS is proposed to deal with a *mobile adversary* [80] who collects the shares of an increasing number of players over time in order to recover the secret. In a PSS, the shares of the players must be updated periodically without changing the secret. This "share update" is done by adding some polynomials $g_i(x)$ with zero constant terms to the original secret sharing polynomial $f(x)$, which has constant term $\alpha$. As a result, the new secret sharing polynomial will be $\hat{f}(x) = f(x) + \sum_i g_i(x)$, and therefore the secret remains $\hat{f}(0) = \alpha$. The share update phase can be done with or without a dealer. Proactive secret sharing schemes can also tolerate a malicious dealer.

We should stress that PSS is different from a *dynamic threshold scheme*, where the parameters of the protocol (such as the threshold, the number of players, etc) can be changed after the scheme's initialization. Next, an example of proactive secret sharing is provided.

**Example 1.3** *Suppose the secret sharing polynomial is $f(x) = \boldsymbol{3} + 4x + 7x^2 + 5x^3 \in \mathbb{Z}_{13}[x]$. Players $P_1, P_2$, and $P_3$, and $P_4$ receive the following shares from the dealer accordingly.*

$$f(1) = 6, f(2) = 1, f(3) = 5, f(4) = 9.$$

*The players securely generate $g(x) = \boldsymbol{0} + 4x + 2x^2 + 10x^3$ in the absence of the dealer with the following shares:*

$$g(1) = 3, g(2) = 5, g(3) = 1, g(4) = 12.$$

*Each player $P_i$ locally adds his shares together. As a result, the new polynomial will be $\hat{f}(x) = \boldsymbol{3} + 8x + 9x^2 + 2x^3$ with the following shares:*

$$\hat{f}(1) = 9, \hat{f}(2) = 6, \hat{f}(3) = 6, \hat{f}(4) = 8.$$

To assign multiple shares rather than a single share to some players in a threshold secret sharing scheme, *weighted secret sharing* (*WSS*) is introduced [10]. For instance, consider the scenario in which the president and chief executive of a company have the collective authority to open the safe deposit box of the company, but that any two vice-presidents can substitute for a missing party in their absence. In this scenario, the weighted scheme is used to prioritize different players.

### 1.3.3 Review of a Simple VSS Scheme

The paper [98] proposes a verifiable secret sharing scheme in which players are able to check the consistency of their shares by symmetric bivariate polynomials. This protocol uses pairwise channels as well as a broadcast channel, and it is secure under the assumption that $|\nabla| \le t-1 \le \lfloor \frac{n-1}{4} \rfloor$ where $\nabla$ denotes the set of bad players.

The reason behind this assumption is that a dishonest dealer may disrupt at most $n/4$ shares in the sharing phase (otherwise he is disqualified), and therefore the players who have received these disrupted shares are removed from the scheme in the "sharing" phase. Moreover, at most $1/3$ of the remaining $3n/4$ shares might be disrupted by the colluders in the "recovery" phase. If $|\nabla| \le \lfloor \frac{n-1}{4} \rfloor$, then a suitable error correction technique, e.g., based on a Reed-Solomon code [58], can be used to recover the secret correctly. All computations are done in $\mathbb{Z}_q$ (where $q$ is a prime number) and $\omega$ is a primitive element in this field. This scheme is presented in Figure 1.2.

In this VSS scheme, the following properties are satisfied (see [98]):

1. If a good player $P_i$ outputs $ver_i = 0$ at the end of the "sharing" phase, then every good player outputs $ver_i = 0$. If this occurs, then more than $t-1$ shares have been corrupted by bad players and a dishonest dealer. In this case, the protocol is stopped.
2. If the dealer is honest, then $ver_i = 1$ for every good $P_i$ at the end of the "sharing" phase. In this situation, at most $t-1$ shares might be later corrupted by bad players.
3. If at least $n-(t-1)$ players $P_i$ output $ver_i = 1$, then $\alpha' \in \mathbb{Z}_q$ will be reconstructed in the "recovery" phase (i.e., at most $t-1$ players have received incorrect shares from the dealer), and $\alpha' = \alpha$ if the dealer is honest.
4. If $|\mathbb{Z}| = q$, $\alpha$ is chosen randomly from $\mathbb{Z}_q$, and the dealer is honest, then any coalition of at most $t-1$ players cannot guess the value $\alpha$ with probability greater than $\frac{1}{q}$ at the end of the "sharing" phase.

Secret Sharing

1. The dealer $D$ selects a symmetric polynomial $f(x, y) \in \mathbb{Z}_q[x, y]$ of degree $t-1$ where $f(0,0)$ is the secret $\alpha$. He sends $f_i(x) = f(x, \omega^i)$ to $P_i$ for $1 \leq i \leq n$.

2. $P_i$ and $P_j$ perform *pairwise checks*; i.e., they verify that $f_i(\omega^j) = f_j(\omega^i)$. If $P_i$ finds that $f_i(\omega^j) \neq f_j(\omega^i)$, he broadcasts the ordered pair $(i, j)$ to accuse $P_j$.

3. Each $P_i$ computes a subset $\Gamma \subseteq \{1, ..., n\}$ such that any ordered pair $(i, j) \in \Gamma \times \Gamma$ is not broadcasted. If $|\Gamma| \geq n-(t-1)$, $P_i$ outputs $ver_i = 1$, otherwise, $ver_i = 0$.

4. The sharing is accepted if at least $n-(t-1)$ players output $ver_i = 1$, otherwise, $D$ is disqualified; $\Gamma$ can be constructed by a polynomial time algorithm [35].

Secret Recovery

1. Each player $P_i$ sends the constant term of his share, that is, $f_i(0)$, to a selected player $P_j$.

2. Player $P_j$ computes a polynomial $f_j(y)$ such that $f_j(\omega^i) = f_i(0)$ for at least $n-2(t-1)$ values of $i$. He then computes the secret $f_j(0) = f(0,0)$.

Figure 1.2: An Unconditionally Secure Verifiable Secret Sharing Scheme

To change this protocol into a *proactive verifiable secret sharing* scheme, we require assumption $|\nabla| < t-1$ as opposed to $|\nabla| \leq t-1$; similarly $|\nabla|$ denotes the number of corrupted parties. Achieving proactivity has three steps:

1. *Renewal*: the servers update their shares by polynomials with zero constant terms.

2. *Detection*: subsequently, the corrupted servers are detected and they are rebooted.

3. *Recovery*: in the final stage, new shares are generated for the rebooted servers.

The renewal phase is shown in Figure 1.3 where the players use symmetric bivariate polynomials with zero constant terms in order to update their shares. For detailed discussions regarding the detection and recovery steps, see [26].

Share Update

1. Each $P_l$ acts as a dealer and selects a random symmetric polynomial of degree $t-2$. He sends $g^l_i(x) = g^l(x, i)$ to $P_i$ for $1 \leq i \neq l \leq n$ through a secure channel.

$$g^l(x, y) = \sum_{i=0}^{t-2} \sum_{j=0}^{t-2} g^l_{ij} x^i y^j \text{ where } g^l_{ij} = g^l_{ji} \text{ for all } i, j.$$

2. To verify the shares distributed by $P_l$, each pair of other players $P_i$ and $P_j$ perform pairwise checks $g^l_i(j) = g^l_j(i)$ through secure channels.

3. If player $P_j$ finds that the equality does not hold for more than $t-2$ values of $i$ (his share $g^l_j(x)$ is not consistent with other shares), player $P_j$ broadcasts an accusation of player $P_l$.

4. If $P_l$ is accused by at most $t-2$ players (otherwise $P_l$ is definitely a bad player), he defends himself by broadcasting $g^l_j(x)$ that he had sent to the accusers.

5. Other players $P_i$, excluding the conflicting parties $P_l$, $P_j$, check $g^l_j(i) = g^l_i(j)$ and broadcast "yes" or "no". If, for each broadcasted share, at least $n - t + 1$ players broadcast "yes", $P_l$ is not guilty and $P_j$ stores the broadcasted $g^l_j(x)$.

6. Finally, each $P_i$ updates the list of good players $\Gamma$ who have not been found guilty in the previous step, and then he updates his shares as follows:

$$f_i(x) = f_i(x) + (x + i) \sum_{l \in \Gamma} g^l_i(x)$$

Figure 1.3: Share Renewal in a Proactive Verifiable Secret Sharing Scheme

## 1.4 Multiparty Computation

*Secure multiparty computation* (*MPC*) [37, 9, 23] was first motivated by the *millionaires' problem* [108]. In this problem, the goal is to determine whether $x > y$, where both $x$ and $y$ are private secrets of two players. The answer to this question becomes known to the two parties after the execution of the protocol.

In a multiparty computation protocol, various users, intelligent agents, or computer servers cooperate in order to perform computations based on the private data they each provide [29], e.g., they compute a function $f(\alpha_1, \ldots, \alpha_n)$, where $\alpha_1, \ldots . \alpha_n$ are private values and $f(\alpha_1, \ldots, \alpha_n) = \alpha$ is revealed to everyone. Since these computations could involve untrusted participants or competitors, consequently, the privacy of each player's input is an important factor. As stated in the literature [38], a fundamental method used in secure multiparty computations is secret sharing.

We present a high-level description of a MPC protocol where the computation phase involves a boolean or arithmetic circuit [9]. Multiparty computation consists of the following three phases: *sharing*, *computation* and *reconstruction*, as shown in Figure 1.4. In the "sharing" phase, each player acts as a dealer to distribute shares of his secret among all the other parties. Subsequently, "computation" is done using two secure operations: *addition* and *multiplication*. Suppose we only have two secrets. In the case of addition, each player locally adds shares of the two secrets. For multiplication, players locally multiply shares of the two secrets by each other. They then collaborate and perform some further computations, which we shortly explain. Finally, in the "reconstruction" phase, players reveal the shares of the function value so that all the players learn it.

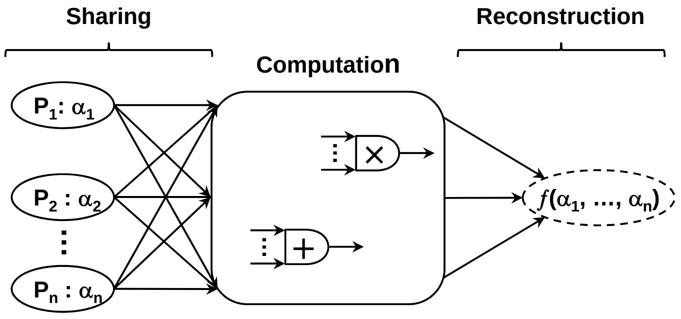

Figure 1.4: Secure Multiparty Computation

Sample applications of such schemes are: *interval test, comparison, and equality test* [25, 69, 68, 33], *joint signature or decryption*, where a group of players sign documents or decrypt messages with the intention that, only if all of them or a specified subset of participants cooperate, then a signature or a message can be generated [38], *shared RSA keys*, in which a number of players collaborate to jointly construct an RSA key [15], and *electronic auctions with private bids*, where a group of agents perform sealed-bid electronic auctions while preserving the privacy of the submitted bids [42]. The main challenge in a secure multiparty protocol protocol is the "multiplication" operation. In the next section, we briefly review an initial solution to this problem.

### 1.4.1 Secure Multiplication of Secrets

Ben-Or et al. [9] proposed a method for the secure multiplication of two secrets. Suppose secrets $\alpha$ and $\beta$ are encoded by two polynomials $f(x)$ and $g(x)$ of degree $t-1$, and each player $P_i$ holds one share on each of these polynomials, $f(i)$ and $g(i)$ respectively. The product of these two secrets $\alpha\beta$ is the constant term of the polynomial $h(x) = f(x) \times g(x)$. If each player multiplies his shares together, the resulting value is a point on $h(x)$. There are two problems with this approach:

1. The degree of $h(x)$ is $2t-2$ instead of the desired $t-1$; i.e., the threshold is increased.
2. Since $h(x) = f(x) \times g(x)$, $h(x)$ is reducible and so it is not a random polynomial.

To overcome these problems, the authors in [9] use a "degree reduction" protocol in which the polynomial $h(x)$ is truncated to decrease its degree to $t-1$. Let $k(x)$ be the resulting truncation of $h(x)$ padded with 0's up to degree $2t-2$. They then apply a simple procedure to randomize coefficients of $k(x)$, except the constant term which remains the product of the two secrets. For the sake of simplicity, suppose $n = 2t-1$, where $n$ is the number of players.

- Let $\vec{H}_{1\times n}$ be the coefficient vector of $h(x)$, and
- let $\mathcal{Z}_{n\times n}$ be a projection matrix, i.e., $\mathcal{Z}_{ij} = 1$ if $i = j$ and $i, j \leq t$, otherwise $\mathcal{Z}_{ij} = 0$.

We therefore have:

$$\vec{K} = \vec{H} \cdot \mathcal{Z}, \text{ where} \tag{1.4}$$

- $\vec{K}_{1\times n}$ is the coefficient vector of $k(x)$, padded with 0's.
- Let $\mathcal{V}'_{n\times n}$ be the transpose of a Vandermonde matrix.

We therefore have:

$$\vec{S} = \vec{H} \cdot \mathcal{V}' \tag{1.5}$$

$$\vec{R} = \vec{K} \cdot \mathcal{V}', \text{ where} \tag{1.6}$$

- $\vec{S}_{1\times n}$ is an evaluation vector of $h(x)$, that is, the players' shares on $h(x)$, and
- $\vec{R}_{1\times n}$ is an evaluation vector of $k(x)$, that is, the players' shares on $k(x)$.

We define $\mathcal{W} = (\mathcal{V}'^{-1} \cdot \mathcal{Z} \cdot \mathcal{V}')$ as a publicly known matrix. Therefore, we have:

$$\begin{aligned}
\vec{S} \cdot \mathcal{W} &= \vec{S} \cdot \mathcal{V}'^{-1} \cdot \mathcal{Z} \cdot \mathcal{V}' \\
&= \vec{H} \cdot \mathcal{V}' \cdot \mathcal{V}'^{-1} \cdot \mathcal{Z} \cdot \mathcal{V}' && \text{by (1.5)} \\
&= \vec{H} \cdot \mathcal{Z} \cdot \mathcal{V}' \\
&= \vec{K} \cdot \mathcal{V}' && \text{by (1.4)} \\
&= \vec{R} && \text{by (1.6).}
\end{aligned}$$

This computation shows that if we multiply the evaluation vector $\vec{S}$ of a polynomial $h(x)$ by $\mathcal{W}$ (as a publicly known matrix), we get the evaluation vector $\vec{R}$ of $h(x)$'s truncation, denoted by $k(x)$. This means, if each player re-shares his share by another polynomial, then each party is going to hold a vector of shares. Subsequently, if the players locally multiply their vectors by $\mathcal{W}$, all the shares are transformed from $h(x)$ of degree $2t-2$ to a new polynomial $k(x)$ of degree $t-1$.

To randomize the coefficients of $k(x)$, players perform the following procedure (that is similar to the "share update" phase of a proactive secret sharing scheme): Each player $P_i$ ($1 \le i \le n$) randomly selects a polynomial $q_i(x)$ of degree $2t-2$ with a zero constant term (the protocol in [9] ensures that malicious players are not able to send shares on a polynomial with non-zero constant term). After that, they each send $q_i(j)$ to $P_j$ for $1 \le j \le n$. Finally, players add these values to their shares on $k(x)$:

$$k(x) = k(x) + \sum_{i=1}^{n} \sum_{j=1}^{n} q_i(j)$$

where $k(0) = \alpha$. We should note that this method of "degree reduction and randomization" was later simplified by Gennaro et al. [36], which is discussed in detail in Section 4.3.2.

## 1.5 Reputation Management

In the context of the social networks, *trust* is the expectation that a player has about the future behavior of another player based on the history of their interactions. On the other hand, *reputation* is the perception that a player creates by past behavior about his intentions. The former is a personal quantity while the latter is a social quantity [65]. A comprehensive survey of existing trust and reputation systems are presented in [47].

To quantify the reputation of each player in a social secret sharing scheme, we apply the trust calculation method proposed in [70] (which is the modified version of the solution in [111]). For additional discussion and motivation regarding this trust function, see [70]. We start with the following definition:

**Definition 1.4** *Let $\mathcal{T}_i^j(p)$ be the* trust value *assigned by player $P_j$ to player $P_i$ in time interval $p$. $\mathcal{T}_i : \mathbb{N} \mapsto \mathbb{R}$ is the* reputation function *used to compute the reputation of $P_i$:*

$$\mathcal{T}_i(p) = \frac{1}{n-1} \sum_{j \neq i} \mathcal{T}_i^j(p),$$

*where $-1 \leq \mathcal{T}_i(p) \leq +1$ and $\mathcal{T}_i(0) = 0$. That is, we calculate the average of the $n-1$ trust values (personal quantities) to compute a player's reputation (a social quantity) [76].*

For instance, let the trust values of $P_1, P_2, P_3$ with respect to $P_4$ be $\mathcal{T}_4^1(p) = 0.4$, $\mathcal{T}_4^2(p) = 0.5$, and $\mathcal{T}_4^3(p) = 0.6$. As a result, reputation of $P_4$ will be $\mathcal{T}_4(p) = 0.5$. In this thesis, only a public value $\mathcal{T}_i(p)$ is assigned to each player $P_i$, where $\mathcal{T}_i(p)$ represents his reputation. This value depends on players' public actions and behavior. Therefore, we can assume that $\mathcal{T}_i(p) = \mathcal{T}_i^j(p)$ for all $j$.

We now briefly review the approach proposed in [70]. As shown in Table 1.1, three "types" of players (that is, $\mathcal{B}$: bad; $\mathcal{N}$: new; and $\mathcal{G}$: good) with six possible outcomes are defined, where $\alpha$ and $\beta$ determine boundaries on the trust values used to define the different sets of players. This approach then applies functions $\mu(x)$ and $\mu'(x)$ respectively to update the reputation of each player $P_i$, as shown in Figure 1.5. Parameters $\eta$, $\theta$, $\kappa$, and $\epsilon$ are used to increment and/or decrement the trust value of a player. In intervals $[1-\epsilon, +1]$ and $[-1, \epsilon - 1]$, functions $\mu(x)$ and $\mu'(x)$ both converge to 0.

We should stress that our trust function is not just a function of a single round, but of the "players' history". That is, it rewards more the better a participant has been, e.g., see Figure 1.5: Cooperation, where $\mathcal{T}_i(p) \in [\alpha, 1-\epsilon]$, and it penalizes more the worse a participant has been, e.g., see Figure 1.5: Defection, where $\mathcal{T}_i(p) \in [\epsilon - 1, \beta]$. In addition, it provides opportunities for newcomers to increase their trust values even where we do not know much about their behavior, e.g., see Figure 1.5: where $\mathcal{T}_i(p) \in [\beta, \alpha]$.

| **Current Trust Value** | **Cooperation** | **Defection** |
|---|---|---|
| $P_i \in \mathcal{B} \quad if \quad \mathcal{T}_i(p) \in [-1, \beta)$ | Encourage | Penalize |
| $P_i \in \mathcal{N} \quad if \quad \mathcal{T}_i(p) \in [\beta, \alpha]$ | Give a Chance | Take a Chance |
| $P_i \in \mathcal{G} \quad if \quad \mathcal{T}_i(p) \in (\alpha, +1]$ | Reward | Discourage |

Table 1.1: Six Possible Actions for the Trust Management

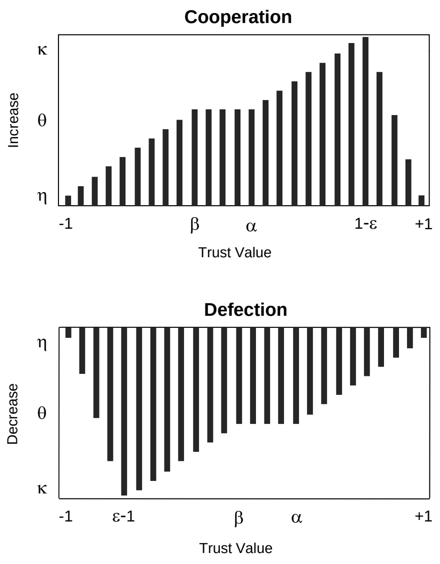

Figure 1.5: Trust Adjustment by $\mu(x)$ and $\mu'(x)$ Functions

Let $\ell_i \in \{0, 1\}$ where $\ell_i = 1$ denotes that player $P_i$ has cooperated in the current period and $\ell_i = 0$ denotes that he has defected. Let $0 < \eta < \theta < \kappa \leq \epsilon$. The proposed trust function is as follows, where $x = \mathcal{T}_i(p-1)$ (i.e., $x$ is the previous trust value):

$$\ell_i = 1 \quad \Rightarrow \quad \mathcal{T}_i(p) = \mathcal{T}_i(p-1) + \mu(x), \text{ where}$$

$$\mu(x) = \begin{cases} \dfrac{\theta - \eta}{\beta + 1}(x+1) + \eta & P_i \in \mathcal{B} \\ \theta & P_i \in \mathcal{N} \\ \dfrac{\kappa - \theta}{1 - \epsilon - \alpha}(x - \alpha) + \theta & P_i \in \mathcal{G}, \mathcal{T}_i(p) \leq 1 - \epsilon \\ \dfrac{\kappa}{\epsilon}(1 - x - \epsilon) + \kappa & \mathcal{T}_i(p) > 1 - \epsilon \end{cases}$$

$$\ell_i = 0 \quad \Rightarrow \quad \mathcal{T}_i(p) = \mathcal{T}_i(p-1) - \mu'(x), \text{ where}$$

$$\mu'(x) = \begin{cases} \dfrac{\kappa}{\epsilon}(x+1) & \mathcal{T}_i(p) < \epsilon - 1 \\ \dfrac{\theta - \kappa}{\beta - \epsilon + 1}(x - \epsilon + 1) + \kappa & P_i \in \mathcal{B}, \mathcal{T}_i(p) \geq \epsilon - 1 \\ \theta & P_i \in \mathcal{N} \\ \dfrac{\eta - \theta}{1 - \alpha}(x - \alpha) + \theta & P_i \in \mathcal{G} \end{cases}$$

Each function $\mu(x)$ and $\mu'(x)$ consists of four linear equations, each of which is simply determined by two points $(x_1, y_1)$ and $(x_2, y_2)$ as follows:

$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1.$$

To ensure that $\mathcal{T}_i(p-1) + \mu(x) \leq 1$ and $\mathcal{T}_i(p-1) - \mu'(x) \geq -1$ when $x = 1 - \epsilon$ and $x = \epsilon - 1$ respectively, we must satisfy the conditions $1 - \epsilon + \kappa \leq 1$ and $\epsilon - 1 - \kappa \geq -1$, or equivalently $\kappa \leq \epsilon$. Because $0 < \eta < \theta < \kappa \leq \epsilon$, it can be shown that $-1 \leq \mathcal{T}_i(p) \leq +1$.

It is worth mentioning that the authors in [70] also define an additional parameter as the *transaction cost* to deal with *cheap* cooperations and *expensive* defections. For instance, consider a scenario in which a player cooperates in regular transactions for several times in order to gain a high trust value. He can then defect in a critical transaction to severely damage the scheme. By considering this transaction cost parameter, a weight for "cooperation" or "defection" is defined and accordingly the trust value is adjusted.

### 1.5.1 New Trust Function

As an alternative solution, we would like to propose a new trust function with "social characteristics", where we consider the other players' behavior. The function that we reviewed in the previous section uses the following two properties in order to adjust the trust value in different cases:

1. *Type*: parameters $\alpha$ and $\beta$ are used to categorize the players in three sets $\mathcal{B}, \mathcal{N}, \mathcal{G}$. Accordingly, six scenarios are considered to increment or decrement the trust value, as shown in Table 1.1.

2. *History*: The trust value $\mathcal{T}_i(p)$ represents a history of actions taken by a player $P_i$ so far. For instance, the "quality" of being a good player ranges from $\alpha$ all the way to $+1$, which quantifies how good a player is and what kind of history the player has.

The previous trust function applies an *individual evaluation strategy.* We intend to define a *social evaluation strategy* by adding the following property to our new trust function:

3. *Sociality*: this takes into account the social behavior of the other parties. In other words, besides players' types and histories, this function considers all players together for trust computation, as opposed to an individual evaluation technique.

As before, $\ell_i = 1$ denotes $P_i$'s cooperation and $\ell_i = 0$ denotes $P_i$'s defection in a specific time interval. Therefore, $\delta = \sum_{i=1}^{n} \ell_i$ denotes the total number of cooperative players in that time interval. Intuitively, the new function should satisfy the following "social" conditions:

1. if $\delta = n$, i.e., if all the players have cooperated, then it is not required to increase the trust value of anyone.
2. if $\delta = 0$, i.e., if all the players have defected, then it is not required to decrease the trust value of anyone.
3. if $\delta > \frac{n}{2}$, i.e., if the majority of the players have cooperated, then cooperation should be rewarded less and defection should be penalized more.
4. if $\delta < \frac{n}{2}$, i.e., if the majority of the players have defected, then defection should be penalized less and cooperation should be rewarded more.
5. if $\delta = \frac{n}{2}$, i.e., if the number of cooperative players and non-cooperative ones are equal, then cooperation and defection should be rewarded and penalized equally.

Our modified trust function, termed *"social trust function"*, is defined as follows, using the previously defined $\mu(x)$ and $\mu'(x)$ functions:

$$\mathcal{T}_i(p) = \begin{cases} \mathcal{T}_i(p-1) + \big(1 - \dfrac{\delta}{n}\big)\mu(x) & \text{if } \ell_i = 1 \\ \\ \mathcal{T}_i(p-1) - \big(\dfrac{\delta}{n}\big)\mu'(x) & \text{if } \ell_i = 0 \end{cases}$$

where $\delta = \sum_{i=1}^{n} \ell_i$. By using the same $\mu(x)$ function for trust increase and reduction in the case of cooperation and defection, the trust function can be simplified as follows:

$$\mathcal{T}_i(p) = \mathcal{T}_i(p-1) + \big(\ell_i - \frac{\delta}{n}\big)\mu(x).$$

An example of the new social trust function is provided in Table 1.2 for further clarification. Each time the players "gain" a portion of the previously defined $\mu(x)$ (e.g., 25%), where this percentage is proportional to the number of "non-cooperative" players. On the other hand, they "lose" a portion of the previously defined $\mu'(x)$ (e.g., 75%), where this percentage is proportional to the number of "cooperative" players.

| $\delta = \sum_{i=1}^{n} \ell_i$ | **Cooperation** | **Defection** |
|---|---|---|
| $n$ | $\mathcal{T}_i(p-1)$ | no defection |
| $\frac{3}{4}n$ | $\mathcal{T}_i(p-1) + 0.25\mu(x)$ | $\mathcal{T}_i(p-1) - 0.75\mu'(x)$ |
| $\frac{1}{2}n$ | $\mathcal{T}_i(p-1) + 0.5\mu(x)$ | $\mathcal{T}_i(p-1) - 0.5\mu'(x)$ |
| $\frac{1}{4}n$ | $\mathcal{T}_i(p-1) + 0.75\mu(x)$ | $\mathcal{T}_i(p-1) - 0.25\mu'(x)$ |
| $0$ | no cooperation | $\mathcal{T}_i(p-1)$ |

Table 1.2: Computing $\mathcal{T}_i(p)$ with Different Values of $\delta$

As stated earlier, reputation is a social quantity representing a player's type and history. Therefore, it is reasonable to assume that "cooperation" has more value if the majority of the players are defecting and vice versa. This is similar to human social life in which cooperation is appreciated more when most of the people are not cooperating. Intuitively, the same justification is true for the case of "defection".

Furthermore, if all the players are cooperating or they all are defecting, their trust values should remain unchanged, no matter what types of players with what kinds of histories are in the society. This can be justified by the uniformity of the players' actions, which occurs in a social situation where there is no competition.

# Chapter 2

# Social Secret Sharing

We introduce the notion of *social secret sharing* (*SSS*), in which shares are allocated based on a player's reputation and the way he interacts with other participants. During the *social tuning phase*, weights of players are adjusted such that participants who cooperate will end up with more shares than those who defect. Alternatively, newcomers are able to be enrolled in the scheme while corrupted players are disenrolled immediately. This scheme proactively renews shares at each cycle without changing the secret, and allows trusted participants to gain more authority. In the proposed schemes, both the passive and active mobile adversaries are considered in an unconditionally secure setting.

## 2.1 Introduction

In this chapter, we first propose a new secret sharing scheme in a social context. In this new cryptographic primitive, shares are allocated based on each player's reputation and behavior. In fact, player's shares are proactively renewed at each cycle without changing the secret, while allowing the cooperative players to gain more authority. We then illustrate how this construction can be used in cloud computing to create a self-organizing environment. In fact, we show how distributed secure systems using threshold secret sharing can be adjusted automatically based on the resource availability of the cloud providers.

Our scheme is called "social secret sharing" since it can be visualized in terms of players collaborating to recover the secret in a social network, based on their reputations. This is similar to human social life where people share more secrets with those whom they really trust and vice versa. In the literature, there exist other types of dynamic schemes with different properties than our constructions, such as schemes in which one can activate various access structures [14], enroll or disenroll players [112], or change the threshold [102].

### 2.1.1 Motivation

Our motivation is that, in real-world applications, components of a secure system may have different *levels of importance*, which is reflected in the number of shares a player has, as well as *reputation*, which is based on the amount of cooperation with other players for "secret recovery". A good construction should balance these two factors respectively, that is, adjusting the number of shares a player can hold based on his reputation. For instance, assume a major shareholder has been attacked. If the scheme is not re-arranged (i.e., the weight of each player is not changed), 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 reasonable to assign him more shares. Equivalently, this is modifying the access structure in an appropriate way based on the players' behavior.

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 players based on their weights in a secure system. Consequently, revealing the secret will trigger an action. The aim is to monitor participants' behavior over time to regulate players' responsibility. As an example of this scenario, we can refer to real time systems that are subject to operational deadlines. If a server provides his share in a single time slot when it is needed, he is classified as a *cooperative* player in that time period. Otherwise, he is not a reliable player.

### 2.1.2 Contributions

As our main contribution, we first explain a scenario in which this cryptographic primitive can be used to create a self-organizing protocol in the cloud. In fact, we show a distributed system can be reconfigured automatically based on the resource availability of the cloud providers. Subsequently, the formal definition and necessary conditions of a social secret sharing scheme is provided.

Afterwards, a scheme under the passive mobile adversary model is constructed, and the required techniques for weight escalation/reduction based on an existing trust computation model [70] is proposed. In addition, a new tool, called the *enrollment protocol*, is developed for this primary protocol.

Our social tuning phase is dealer-free, unconditional, and secure under the passive or active mobile adversary models. In fact, it is quite challenging to design protocols in this setting. Moreover, if one relaxes any of these requirements, he can then decrease the computation and communication complexities. For instance, by using a trusted authority, or by constructing the proposed schemes in a computational setting, or by considering the simple passive adversary model without mobility, simpler protocols can be constructed.

### 2.1.3 Organization

This chapter is organized as follows: Section 2.2 creates a general picture of our social secret sharing scheme. Section 2.3 provides an application of the proposed scheme. Section 2.4 demonstrates the first construction under the passive mobile adversary model. Section 2.5 describes a solution in the active mobile adversary model. Finally, Section 2.6 contains concluding remarks.

## 2.2 Social Secret Sharing

The proposed social secret sharing model consists of $n$ participants, $P_1, P_2, \ldots, P_n$, and a dealer who is available only during the initialization phase. We assume the existence of private channels between each pair of participants (to be used during the share renewal step), and we assume that the dealer can communicate securely with participants in the dealing stage. We also assume the existence of a synchronized broadcast channel, on which information is transmitted instantly and accurately to all participants.

Let $\mathcal{M}_{n\times m}$ be a matrix representing the players' identifiers, where $n$ is the maximum number of players, $m$ is the maximum weight of each player, and the entries in $\mathcal{M}_{n\times m}$ are distinct non-zero elements of $\mathbb{Z}_q$. For instance, in the case of $\mathcal{M}_{2\times 2}$, then entries $m_{11}$ and $m_{12}$ are the identifiers of player $P_1$ and entries $m_{21}$ and $m_{22}$ are the identifiers of $P_2$. Let $\mathbb{Z}_q$ be a finite field and let $\omega$ be a primitive element in this field. All computations are performed in the finite field $\mathbb{Z}_q$.

Our intention is to construct unconditionally secure schemes, i.e., schemes that do not rely on computational assumptions. We consider both passive and active adversaries with mobility, i.e., adversaries who are able to change the set of corrupted players from time to time during the execution of protocols. In our first construction, we assume that players correctly follow all protocols but are curious to learn the secret, while in the second one, we assume that players may deviate from the protocols.

In social secret sharing, each participant initially receives a constant number of shares. As time passes, players are reassigned weights based on their behavior in the scheme. Consequently, each participant receives a number of shares corresponding to his trust value, which is the representation of a player's current reputation. (Recall that reputation is based on the availability of the players and cooperative players are available more often.) Weights of participants are adjusted such that cooperative players receive more shares than non-cooperative ones. The motivation here is that this increases the availability of the system over time.

In addition, newcomers can join the scheme while corrupted players are disenrolled. The reason for a corruption might be an active attack or a computational failure. A corrupted server can return to the scheme (as a newcomer) only after being fixed.

To further illustrate the proposed scheme, the required assumptions are first defined in order to ensure that the scheme works correctly. Subsequently, different possible behaviors are defined. Finally, the formal definition of a social secret sharing scheme is illustrated. The following assumptions are required to construct a social secret sharing scheme:

1. To recover the secret, the total weight of authorized players $P_i \in \Delta$ must be equal or greater than the threshold, i.e., $\sum_{P_i \in \Delta} w_i \geq t$, where $\Delta$ denotes the set of uncorrupted participants. We later show that this set $\Delta$ is further divided into three subsets $\mathcal{B}$*: bad,* $\mathcal{N}$*: new,* and $\mathcal{G}$*: good* that represent *non-cooperative, new,* and *cooperative* players respectively.

2. On the other hand, the total weight of colluders $P_i \in \nabla$ must be less than the threshold, where $\nabla$ denotes the set of corrupted players, i.e., $\sum_{P_i \in \nabla} w_i < t$.

3. Finally, the weight of each $P_i$ is bounded by a parameter $m$ which is much less than $t$, that is, $w_i \leq m \ll t$ for $1 \leq i \leq n$.

**Definition 2.1** *The following* actions *are defined, where a player's action* $a_i \in \{\mathcal{C}, \mathcal{D}, \mathcal{X}\}$*.*

- $\mathcal{C}$ *denotes* cooperation*, where player* $P_i$ *is available at the required time and he sends correct shares to the other parties.*

- $\mathcal{D}$ *denotes* defection*, where player* $P_i$ *is not available at the required time or he responds with a significant delay.*

- $\mathcal{X}$ *denotes* corruption*, where player* $P_i$ *has been compromised by an adversary and he may send incorrect shares.*

A *social secret sharing* scheme consists of three phases: secret sharing, social tuning, and secret recovery. The only difference compared to a traditional threshold scheme is the social tuning stage, in which the weight of each player $P_i$ is adjusted based on the player's reputation $\mathcal{T}_i(p)$. After that, shares are updated accordingly. The details of this procedure will be discussed in later sections of this chapter.

### 2.2.1 Social Tuning in a Nutshell

In a social secret sharing scheme, players first receive multiple shares from the dealer. Subsequently, the scheme is readjusted based on the players' behavior. We review the social tuning phase that consists of the following steps. For the sake of simplicity, suppose we increase or decrease the weights of the players one by one.

1. *Adjustment*: based on the players' availability or response time, the "reputation" and consequently the "weights" of all the players are adjusted.
2. *Enrollment*: to increase the weight of a cooperative player, say by one, parties jointly collaborate to generate a new share on the original secret sharing polynomial $f(x)$ for the cooperative player. This procedure is done in the absence of the dealer. For details, see the *enrollment* protocol in [76].
3. *Disenrollment*: to decrease the weight of a player, say by one, parties jointly collaborate to update all shares except one share of the non-cooperative player. That is, while all shares are updated to be on a new secret sharing polynomial $\hat{f}(x)$, one share remains on $f(x)$, and as a result, the share will not be valid anymore.

As shown in Figure 2.1, share $f(s_{14})$ is first enrolled and then share $f(s_4)$ is disenrolled.

## 2.3 Application: Self-Organizing Clouds

In *cloud computing*, different commercial providers (such as Amazon, Google, and Microsoft) offer computing services to consumers. The major goal is to provide computing, storage, and software as a service. As a result, consumers do not need to invest in IT infrastructure of their own. They can obtain these services from external providers according to their demands by a pay-per-use model [106], i.e., obtaining more services in the case of growing demand and vice versa.

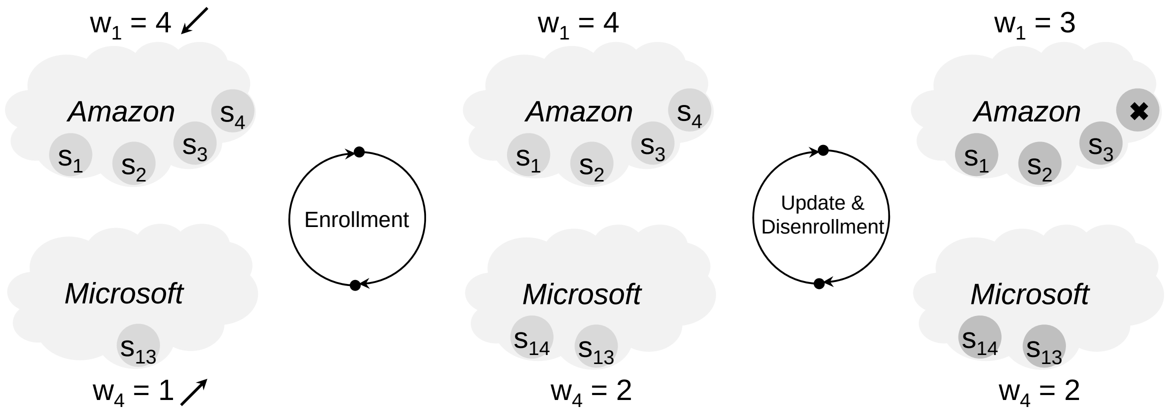

Figure 2.1: Social Secret Sharing Scheme

A significant challenge in cloud computing is resource management, due to consumers' expectations in terms of resource availability, overall performance, etc. In some settings, enterprises provide valuations to service providers (i.e., the money they are going to pay if cloud providers satisfy their demands). The service providers then try to maximize their own profit, for instance, by prioritizing the consumers' jobs. All these factors may lead to competition, negotiation, dynamic allocation, and automatic load balancing. For an extensive survey on this matter, see [21].

We demonstrate a new method of share distribution over the cloud in a secure system using threshold secret sharing. The question is how such systems can be automatically configured based on the availability of different components. This can help to better comply with the *service-level agreements* (*SLA*) established between the cloud providers and consumers. We believe that the challenge can be seen as a cooperative game between the cloud providers and consumers, that is:

1. For the service providers to comply with the service-level agreements, and
2. for the consumers to receive services with a high satisfaction rate.

As an example, we can refer to a significant spike in online shopping with "Amazon" at the end of the year. It would be helpful for both consumers and service providers if the system adopts an automatic reconfiguration strategy and relies less on busier components during certain periods. We illustrate how this can be accomplished by regular interactions between the providers and consumers.

The good news is that, in a distributed secure system using threshold secret sharing, even if some servers do not act properly (for instance, due to an adversarial attack or delay in response time), the system can still accomplish the task if a certain number of components operate appropriately. Therefore, we intend to show this cooperation can be modeled by social secret sharing. In other words, the consumers use a reputation management system to rate different components of the cloud. Subsequently, the system is reconfigured over the cloud to guarantee the service-level agreement.

Our model consists of a dealer who initiates a weighed secret sharing scheme, $n$ cloud providers denoted by $P_1, \ldots, P_n$, and many servers interacting with the cloud providers. Let $\mathcal{T} = (\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n)$ and $\vec{w} = (w_1, w_2, \ldots, w_n)$ be the vector of players' trust values and the vector of players' weights respectively. The initial values in $\mathcal{T}$ are all zero (that is, all service providers are treated as newcomers), whereas the initial values in $\vec{w}$ are chosen by the dealer based on a specific distribution.

Let us assume a secret key $\alpha$ is selected in order to accomplish a secure task whenever it is required. For instance, we can refer to secure auctions in which bidders submit their sealed-bids to auctioneers when the auction starts and then the auctioneers define the outcomes (i.e., the winner and the selling price) without revealing the losing bids [85].

We can therefore assume that secret key $\alpha$ is used by many auctioneers to start or accomplish several sealed-bid auctions over time on behalf of a seller. Considering this secure auction scenario, a dealer (or a seller) first distributes shares of this secret among different service providers (or clouds) according to their initial weights in vector $\vec{w}$, as shown in Figure 2.2. He then leaves the scheme.

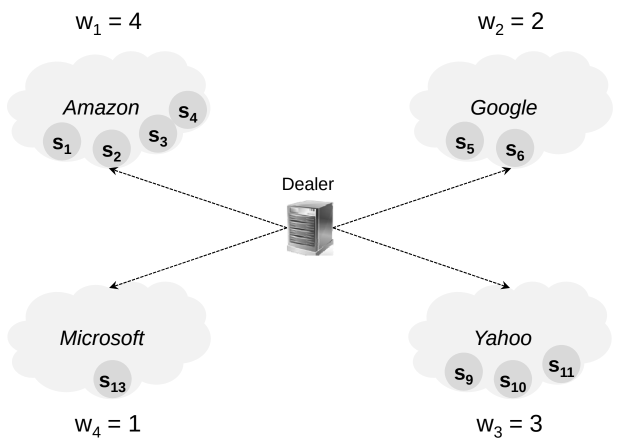

Figure 2.2: System Initialization

Subsequently, different servers (or auctioneers) interact with the cloud providers to perform their tasks in the absence of the dealer; see Figure 2.3. For instance, from time to time, requests for these shares are sent to the cloud providers by the servers. The secret is recovered on these servers and then a secure procedure (or sealed-bid auction) is accomplished. Finally, the secret and its corresponding shares are erased from the servers.

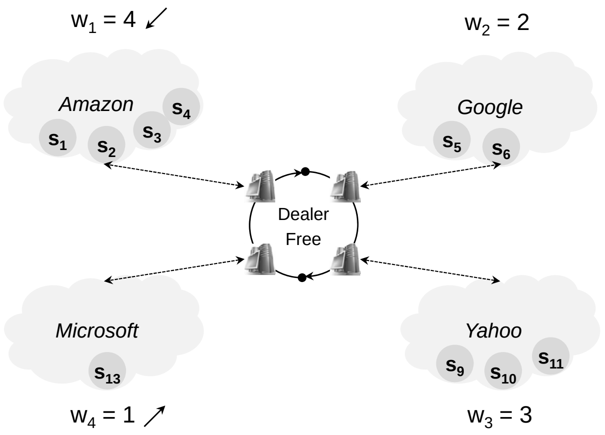

Figure 2.3: Weight Adjustment

Based on the service providers' actions $a_i \in \{\mathcal{C}, \mathcal{D}\}$ (Definition 2.1) as well as a trust function, these servers rate each component of the cloud in terms of its response time; this issue is going to be more critical in real-time systems where "response time" plays an important role. Consequently, the weight of each service provider is changed according to his new trust value. For instance, as shown in Figure 2.3, the weights of two components are going to be updated. To see how increase or reduction of a trust value affects the weight of a player, see [70], for a discussion of *trust-to-share ratio* computation.

In the case of corruption, where $a_i = \mathcal{X}$, the corrupted providers are rebooted. They then return to the scheme and are treated as newcomers. We should note that corrupt actions (for instance, sending incorrect shares to the other players) are detectable by using a verifiable secret sharing scheme.

In the final phase, the service providers jointly collaborate to reconfigure the scheme according to new weights. They initially create the new shares by using an *enrollment* protocol, e.g., the share $f(s_{14})$ is enrolled for "Microsoft". Subsequently, shares are updated (except for the shares that are scheduled to be disenrolled) such that they are transformed to a new secret sharing polynomial, e.g., if the share $f(s_4)$ is not updated, then "Amazon" is going to have three shares afterward, as shown in Figure 2.4. Misbehavior during this phase can be also detected and punished if desired.

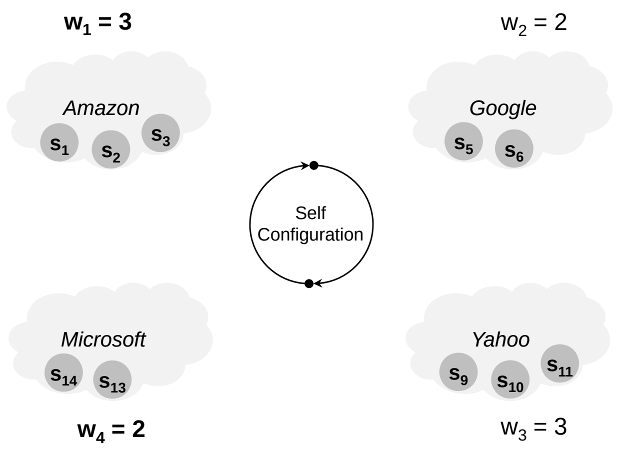

Figure 2.4: Self-Configuration

The benefit of using threshold secret sharing schemes in a distributed secure system is its “fault-tolerance” and “availability”. For instance, if one component is compromised by an adversary or he responds with delay, other parties can carry out the intended procedure.

## 2.4 Passive Adversary Model Construction

In this construction, we consider the *passive adversary model*, where players follow all protocols but an unauthorized subset of them may collude to gather information and attempt to reconstruct the secret.

### 2.4.1 Secret Sharing

Suppose the dealer initiates a secret sharing scheme by generating $f(x) \in \mathbb{Z}_q[x]$ of degree $t-1$ where its constant term is secret $f(0) = \alpha$, i.e., Shamir threshold scheme, as discussed in Section 1.3.1. He sends shares of player $P_i$ for $1 \leq i \leq n$ according to his weight $w_i$, and then he leaves the scheme:

$$\varphi_{ij} = f(\vartheta_{ij}) \text{ for } 1 \leq j \leq w_i$$

where $\vartheta_{ij} = im - m + j$ and $m$ is the maximum weight of any participant, as defined in Section 2.2. The initial trust value is zero for all players. This trust value and, consequently, weights of participants are updated at each cycle during the share renewal stage based on players’ behavior.

### 2.4.2 Social Tuning

Our scheme provides a mechanism for assigning new weights to players based on their behavior at the end of each time period, where by *behavior* we refer to a participant's reputation. We apply a weight adjustment technique that rewards reliable participants because of their repeated cooperation, reduces the influence of unreliable players due to their past defection, and protects the scheme from colluders.

The trust function illustrates how reputable or trustworthy each participant is. One simple solution is to assign an initial trust value to newcomers, increase this value by a constant if the participant is cooperating, and decrease it otherwise. However, this naive method does not consider various scenarios when making the adjustment. For that reason, we apply the proposed trust function presented in Section 1.5.

Although we upper-bounded the weight of each player by $m$, trust function $\mathcal{T}_i(p)$ also bounds the trust value, both above and below, so that a participant cannot continually build up his reputation in order to be the main shareholder and form a monopoly. This protects the scheme in a scenario where a malicious player cooperates for a while to gather most of the shares and then defects.

Finally, since players' weights and consequently their trust values are public information, the trust computation and weight adjustment can be done by any authority or a committee of players on a public board. In the next sections, we illustrate how to increase and/or decrease the weights of different players consistently.

#### a. Inactivating Non-cooperative Players' Shares.

Now that we have a trust value for each participant, we turn to the task of using that value to adjust the scheme. Clearly, some identifiers $j$ $(1 \leq j \leq m)$ should be inactivated for each player whose trust value has been decreased, and activated for players whose trust values have risen, as well as for newcomers. The task is to determine how many, say $\delta$, identifiers should be inactivated for non-cooperative participants, and how $\delta$ new identifiers should be activated for cooperative players and newcomers.

One option for share removal is simply to disenroll a single *id* of a player each time his trust value decreases. However, such an approach does not take into account the total number of shares in the scheme, nor does it consider the number of shares each participant has. For instance, if the player has a large number of shares, inactivation of a single *id* has a small effect. On the other hand, a participant with only one share remaining would be totally removed from the scheme. In this case, one particular approach is to inactivate a number of *ids* for each player $P_i$ that is proportional to the amount that the player's reputation $\mathcal{T}_i(p)$ is decreased:

$$a_i = \mathcal{D} :\Rightarrow w_i(p) = \left\lfloor w_i(p-1) \cdot (1 - \frac{\tau}{2}) \right\rfloor \tag{2.1}$$

where $\tau = \mathcal{T}_i(p-1) - \mathcal{T}_i(p) \geq 0$ denotes the trust value reduction for a non-cooperative player $P_i$. Since the trust values $\mathcal{T}_i(p)$ are in the range $[-1, +1]$ for every player, we divide $\tau$ by 2 in order to compute the rate of weight reduction in the interval $[0, 1]$. If $w_i(p)$ becomes zero, player $P_i$ is removed from the scheme, which results in the release of a row in $\mathcal{M}_{n \times m}$. The total number of *ids* to be deactivated (and then activated) in the entire scheme is given as follows:

$$\delta(p) = \sum_{i \,:\, a_i = \mathcal{D}} \Big( w_i(p-1) - w_i(p) \Big). \tag{2.2}$$

**Example 2.2** *Suppose that trust values of a bad player $P_i$ and a good player $P_j$ have been decreased from $\mathcal{T}_i(p-1) = -0.2$ to $\mathcal{T}_i(p) = -0.8$ and from $\mathcal{T}_j(p-1) = 0.8$ to $\mathcal{T}_j(p) = 0.2$ respectively. As a result, $\tau = 0.6$ would be the same for both players. Therefore, the rate of weight reduction computed in (2.1) would be $0.7$. This means that each player loses approximately $30\%$ of his shares.*

**b. Activating Cooperative Players' Shares.**

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, $\mathcal{T}_i(p)$, to the number of shares he is holding, $w_i(p)$. This ratio $\rho = \mathcal{T}_i(p)/w_i(p)$ increases with an increase in the participant's trust value, and decreases as the participant gains more shares.

As a result, it is reasonable to activate *ids* in participants for whom this ratio is highest, but this is not enough since we also need to consider newcomers, whose trust values are initially defined to be zero. Therefore, to have a fair policy, we give the "first" priority to cooperative players for whom this ratio is both highest and positive, the "second" priority to newcomers, and the "third" priority to other cooperative players with negative trust values. However, the assumptions of social secret sharing, as stated in Section 2.2, must be satisfied in the following Algorithm.

**Algorithm 1** : Activation of Players' *ids*

```
collect the cooperative and new players in an array $\mathcal{R}$
compute the trust-to-share ratio $\rho$ for all $P_i \in \mathcal{R}$
sort the array $\mathcal{R}$ based on the computed ratios $\rho$
\\ for the sake of simplicity, assume $\delta(p) \leq |\mathcal{R}|$

$k := 0$
for $j := 1$ to $|\mathcal{R}|$ do
   select player $P_i$ from $\mathcal{R}[j]$
   if $0 < w_i(p) < t - 1$ then
      \\ an existing player
      activate a new id for $P_i$
      $w_i(p) := w_i(p-1) + 1$
      $k := k + 1$
   else if $w_i(p) := 0$ then
      \\ a new player
      assign a new row in $\mathcal{M}_{n \times m}$ to $P_i$
      activate a new id for $P_i$
      $w_i(p) := 1$
      $k := k + 1$
   end if
   if $k := \delta(p)$ then
      break the loop
   end if
end for
```

We assume that there are enough cooperative and/or new players $P_i$ with $w_i(p) < t-1$ to activate their *ids*, i.e., $\delta(p) \leq |\mathcal{R}|$. 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 non-cooperative participants who still have relatively high trust values; doing so maintains a constant number of shares in the scheme. By sorting the array $\mathcal{R}$ and assuming $|\mathcal{R}| \approx n$, the complexity of the algorithm is $O(n + n \log n) = O(n \log n)$.

To add participants to the scheme, we have two options for assigning *ids* to new players in $\mathcal{M}_{n \times m}$. The first solution is to add a new row for each new player in the matrix. As time passes, this approach leads to a big matrix with empty rows. The second alternative is to reuse released rows of the disenrolled players. Since we first remove players from the scheme and then update shares of remaining participants, we can reuse the released *ids* and assign them to newcomers without leaking any information about the secret. In fact, new players will receive updated shares corresponding to those recycled *ids*.

**c. Share Renewal.**

This stage consists of two phases. First, initial shares for newcomers (or newly activated *ids* of existing players) are generated. Then, players proactively update their shares, while disenrolled *ids* do not receive any more shares. As a result, old shares corresponding to those inactivated *ids* become useless since they remain on an old secret sharing polynomial.

To update shares in a proactive scheme, a participant must have his previous shares. Suppose we intend to activate a new *id* in period $p$, where we do not have the corresponding old share in period $p-1$. For the sake of simplicity, assume each participant has only one identifier. In this case, if at least $t$ participants cooperate, they can generate the old share for the newcomer, where $t$ is the threshold.

The initial solution for generating an old share, named *share recovery*, was proposed in [44]. That solution is not efficient due to its random shuffling procedure. Saxena et al. [93] proposed a non-interactive solution by using bivariate polynomials, named *bivariate admission control*, but this protocol is secure only under the discrete logarithm assumption. Our solution, as shown in Figure 2.5, is called the *enrollment protocol.* It is an efficient new construction with unconditional security under the passive adversary model. We assume that this protocol is executed in a single time slot in our social secret sharing scheme. We also provide a toy example of this scheme.

**Example 2.3** *Assume $t = 3$ and the dealer has generated shares of three players $P_1, P_2$, and $P_3$ based on $f(x) = 9 + 2x + 5x^2 \in \mathbb{Z}_{13}[x]$, i.e., $\varphi_1 = 3$, $\varphi_2 = 7$, and $\varphi_3 = 8$. After some time, players are asked to create a share for a newcomer (for instance $P_4$) in the absence of the dealer. First each player $P_i$ privately computes $\varphi_i \times \gamma_i$ as follows:*

$$\varphi_i \times \gamma_i = \begin{cases} 3 \times \frac{(4-2)(4-3)}{(1-2)(1-3)} = 3 \\ 7 \times \frac{(4-1)(4-3)}{(2-1)(2-3)} = 5 \\ 8 \times \frac{(4-1)(4-2)}{(3-1)(3-2)} = 11 \end{cases} \qquad \mathcal{E}_{t\times t} = \begin{pmatrix} \partial_{11} = 1 & \partial_{21} = 1 & \partial_{31} = 1 \\ \partial_{12} = 2 & \partial_{22} = 1 & \partial_{32} = 2 \\ \partial_{13} = 4 & \partial_{23} = 2 & \partial_{33} = 5 \end{pmatrix}$$

Enrollment Protocol

1. First, each player $P_i$ for $1 \leq i \leq t$ computes his corresponding Lagrange interpolation constant as follows:

$$\gamma_i = \prod_{1 \leq j \leq t, i \neq j} \frac{k-j}{i-j} \tag{2.3}$$

where $i, j$, and $k$ represent players' *ids*.

2. Each player $P_i$ multiplies his share $\varphi_i$ by his Lagrange interpolation constant, and randomly splits the result into $t$ portions in order to exchange them with other parties:

$$\varphi_i \times \gamma_i = \partial_{1i} + \partial_{2i} + \cdots + \partial_{ti} \quad \text{for } 1 \leq i \leq t. \tag{2.4}$$

3. Players exchange $\partial_{ji}$'s accordingly through pairwise channels. Subsequently, each player $P_j$ holds $t$ values. Player $P_j$ adds the $t$ values together and sends the result to player $P_k$.

$$\sigma_j = \sum_{i=1}^{t} \partial_{ji} \tag{2.5}$$

where $\partial_{ji}$ is the $j^{th}$ share-portion of the $i^{th}$ participant.

4. Finally, $P_k$ adds the received values $\sigma_j$ (for $1 \leq j \leq t$) together to compute his share $\varphi_k$:

$$\varphi_k = \sum_{j=1}^{t} \sigma_j. \tag{2.6}$$

Figure 2.5: Enrollment Protocol in the Passive Adversary Model

*After that, they randomly split the results and exchange them, as shown in the share-exchange matrix $\mathcal{E}_{t \times t}$. Players then compute and send $\sigma_1 = 7, \sigma_2 = 4$, and $\sigma_3 = 8$ to $P_4$. Finally, $P_4$ adds up these values to compute his new share $\varphi_4 = 6$.*

**Theorem 2.4** *The enrollment protocol in Figure 2.5 is correct and unconditionally secure under the passive adversary model tolerating $t-1$ colluders.*

**Proof.** We first show the protocol is correct and then prove its unconditional security.

The following computation illustrates that the new value $\varphi_k$ is in fact $P_k$'s share on $f(x)$:

$$\begin{aligned}
\varphi_k &= \sum_{j=1}^{t} \sigma_j && \text{by (2.6)} \\
&= \sum_{j=1}^{t}\sum_{i=1}^{t} \partial_{ji} && \text{by (2.5)} \\
&= \sum_{i=1}^{t}\sum_{j=1}^{t} \partial_{ji} && \\
&= \sum_{i=1}^{t} \big(\varphi_i \times \gamma_i\big) && \text{by (2.4)} \\
&= \sum_{i=1}^{t} \left(\varphi_i \times \prod_{1\le j\le t, i\ne j} \frac{k-j}{i-j}\right) && \text{by (2.3)} \\
&= f(k) && \text{by (1.1).}
\end{aligned}$$

As shown in the enrollment protocol, each player $P_i$ first multiplies his share $\varphi_i$ by the corresponding Lagrange interpolation constant $\gamma_i$, and then splits the result into $t$ pieces. We defined the *share-exchange matrix* $\mathcal{E}_{t\times t}$, where each row presents various pieces of a single share and each column represents portions of different shares that each player receives. In other words, all values in the $i^{th}$ row, i.e., $\partial_{1i}, \partial_{2i}, \cdots, \partial_{ti}$, originate from a single player $P_i$ and the entries in the $j^{th}$ column, i.e., $\partial_{j1}, \partial_{j2}, \cdots, \partial_{jt}$, represent values that player $P_j$ receives from other participants.

$$\mathcal{E}_{t\times t} = \begin{pmatrix} \partial_{11} & \partial_{21} & \cdots & \partial_{t1} \\ \partial_{12} & \partial_{22} & \cdots & \partial_{t2} \\ \vdots & \vdots & \ddots & \vdots \\ \partial_{1t} & \partial_{2t} & \cdots & \partial_{tt} \end{pmatrix}.$$

We consider the following two scenarios to show that a coalition of $t-1$ participants cannot learn any information about the secret.

First, suppose that $t-1$ of $t$ 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 single unknown row, because $t-1$ columns belong to them. Therefore, just one entry remains unknown, but this is sufficient to prevent the colluders from finding the newcomer's share and consequently the secret remains unknown. As presented in Example 2.3, if players $P_1$ and $P_2$ collude, then $\partial_{33} = 5$ in the third row is not known to the colluders.

Second, suppose that $t-2$ of $t$ participants with the newcomer collude. In this case, the colluders have access to all entries of $t-2$ rows. In addition, they also know $t-2$ entries of the two unknown rows, because $t-2$ columns belong to them. Therefore, four entries remain unknown. On the other hand, the newcomer also knows the sum of the column's entries for all columns. As a consequence, he can just construct two equations with four unknowns which does not reveal any information about the secret. As presented in Example 2.3, if $P_1$ and the newcomer $P_4$ collude, then $\partial_{22} = 1$ and $\partial_{32} = 2$ in the second row and $\partial_{23} = 2$ and $\partial_{33} = 5$ in the third row remain unknown and $P_4$ can only construct the following two equations: $1 + \partial_{22} + \partial_{23} = 4$ and $1 + \partial_{32} + \partial_{33} = 8$.

However, for the given set of $t-1$ players to compute the secret, they need to compute a particular linear combination of $t$ shares and these $t$ shares can only be computed if all of the $\partial_{ji}$'s are known. This linear combination involves different Lagrange constants than the ones that we used to construct the $\partial_{ji}$'s in Figure 2.5. Therefore, the desired computation cannot be carried out since these $t$ players do not know all of the $\partial_{ji}$'s. The other thing we need to verify is that it is impossible to compute the desired linear combination without knowing all the $\partial_{ji}$'s. This could be done only if the ratio of two "old" Lagrange coefficients is the same as the ratio of two "new" Lagrange coefficients, which it is possible to prove cannot happen. ∎

In the next phase, shares of the players are updated by polynomials $g^u(x)$ that have zero constant terms, as shown in Figure 2.6. Therefore, secret $\alpha$ remains the same while the shares of the players are updated (to overcome a mobile adversary [44]). As we mentioned earlier, inactivated *ids* do not receive any shares at this stage and they remain on the old secret sharing polynomial, i.e., they are disenrolled.

### 2.4.3 Secret Recovery

As stated earlier, authorized players are able to recover the secret if their total weight is equal or greater than the threshold, i.e., if $\sum_{P_i \in \Delta} w_i \geq t$. In this case, players $P_i \in \Delta$ send their shares $\varphi_{ij}$ for $1 \leq j \leq w_i$ to a selected participant to reconstruct $f(x)$ by Lagrange interpolation, and consequently the secret $f(0) = \alpha$ is recovered.

**Theorem 2.5** *The social secret sharing scheme ($\mathcal{S}ha, \mathcal{T}un, \mathcal{R}ec$) presented in Section 2.4 is unconditionally secure under the passive mobile adversary model.*

Share Update

1. To update shares, each player $P_u$ generates a random polynomial $g^u(x) \in \mathbb{Z}_q[x]$ of degree $t-1$ with a zero constant term.

2. $P_u$ then sends $w_i$ shares $\psi^u_{ij} = g^u(\vartheta_{ij})$, where $1 \leq j \leq w_i$, to $P_i$ for $1 \leq i \leq n$, where $\vartheta_{ij} = im - m + j$ and $m$ is the maximum weight of any participant.

3. Finally, each player $P_i$ updates his share by adding the auxiliary shares $\psi^u_{ij}$ to his share $\varphi_{ij}$:
$$\varphi_{ij} = \varphi_{ij} + \sum_{u=1}^{n} \psi^u_{ij} \text{ for } 1 \leq j \leq w_i.$$

Figure 2.6: Share Update in the Passive Adversary Model

The security of $\mathcal{S}ha$ and $\mathcal{R}ec$ is the same as the security of the Shamir's scheme [94]. The security of $\mathcal{T}un$ depends on the share renewal step that consists of the following two protocols: *enrollment protocol* and *share update*. The enrollment protocol is secure as shown in Theorem 2.4. Share update is also proven to be secure; see [44].

## 2.5 Active Adversary Model Construction

In this section, we consider the active adversary model, where the players may deviate from protocols or collude to reconstruct the secret. We use the *proactive verifiable secret sharing scheme* that we presented in Section 1.3.3. We modify those protocols accordingly to adapt them to our social secret sharing scheme.

As in the previous section, each player $P_i$ has $w_i$ shares where $w_i$ is his current weight. The total number of shares will be denoted by $W$, where $W = \sum_{i=1}^{n} w_i$. We will require that the number of bad shares, denoted by $\xi$, satisfies the inequality $\xi \leq t - 1 < \left\lfloor \frac{W-1}{4} \right\rfloor$. Our protocol is a small modification of the VSS schemes presented in Section 1.3.3. As before, a pairwise check procedure will be used to verify the consistency of all pair of shares. Also, the "recovery protocol" is used to generate new shares for newly activated *ids*. This recovery protocol is used to restore incorrect shares after running the detection procedure. We assume that the share renewal step is instantaneous, therefore, the adversary cannot corrupt additional participants while shares are being updated.

As before, we consider the matrix $\mathcal{M}_{n\times m}$ of the participants' identifiers, where $n$ is the maximum number of players and $m$ is the maximum weight of any player. We also denote $\vartheta_{ij} = im - m + j$ to define the entries of the matrix $\mathcal{M}_{n\times m}$ in our protocols.

The initial secret sharing by the dealer is essentially identical to protocol in Figure 1.2. The only difference is that now we have $W$ shares instead of $n$ shares; pairwise checks are performed on all pairs of shares. In step 3, $\Gamma$ will be a subset of consistent shares. If $|\Gamma| \geq W - (t-1)$, the sharing is going to be accepted. During secret recovery, the constant terms of all the shares should be sent to a selected player.

Social tuning is similar to its counterpart in the passive adversary model (Section 2.4.2). The only difference is the "share renewal" stage. In other words, the players inactivate (or activate) the shares of non-cooperative (or cooperative) players in the same way that we explained in Section 2.4.2.

Share renewal consists of two phases. In the first phase, initial shares for newcomers or newly activated *ids* of existing players are generated, i.e., they are enrolled, as shown in Figure 2.7. Then, in the second phase, players proactively update their shares, while disenrolled *ids* do not receive any updates.

Enrollment Protocol

1. For every share $\varphi_{ik}(x) \in \Gamma$, $\varphi_{ik}(\omega^{\vartheta_{jl}})$ is sent to a selected player $P_j$ in order to generate his $l^{th}$ share, that is, $\varphi_{jl}(x)$.
2. After that, $P_j$ computes a polynomial $\varphi_{jl}(x)$ of degree at most $t-1$ such that $\varphi_{jl}(\omega^{\vartheta_{ik}}) = \varphi_{ik}(\omega^{\vartheta_{jl}})$ for at least $W - 2(t-1)$ indices $ik$.

Figure 2.7: Enrollment Protocol in the Active Adversary Model

In fact, share $\varphi_{jl}(x)$ is constructed through the interpolation of pairs $(\omega^{\vartheta_{ik}}, \varphi_{ik}(\omega^{\vartheta_{jl}}))$ in the second step. We now explain why we need at least $W-2(t-1)$ indices $ik$ to interpolate polynomial $\varphi_{jl}(x)$ in the above protocol. As we earlier stated, the scheme itself can tolerate $t-1$ incorrect shares. In addition, a dishonest dealer may transmit $t-1$ incorrect shares during sharing in order to eliminate them from the scheme. As a result, set $\Gamma$ of consistent shares has size at least $W - 2(t-1)$ shares. Therefore, an error correction technique, such as the one proposed in [87], can be used to interpolate polynomial $\varphi_{jl}(x)$.

Share update phase is done using the protocol in Figure 1.3. As before, $n$ is replaced by $W$ and each player acts as an independent dealer to distribute shares of a new polynomial of degree at most $t-2$. If a player is detected as a bad dealer, he will be removed from the scheme. This can be done through pairwise checks on distributed shares and accusation procedure, as in step 4 of the protocol. Finally, secret recovery is the same as the protocol in Figure 1.3.3.

## 2.6 Conclusion

We introduced the notion of a *social secret sharing scheme*, in which a player's weight is adjusted based on his reputation and behavior over time. We then demonstrated an application of social secret sharing in cloud computing. Finally, we proposed two constructions that are secure in the passive and active mobile adversary models respectively.

By providing a model for resource management in self-organizing clouds, we showed how a new line of research can be opened within cross-interdisciplinary areas. In fact, using various tools from different disciplines (such as cryptography, reputation systems, and cloud computing) can provide a better way to address the challenges of both existing and new models.

Our proposed construction has a variety of desirable properties: it is *unconditionally secure*, meaning that it does not rely on any computational assumptions; *proactive*, refreshing shares at each cycle without changing the secret; *dynamic*, allowing changes to the access structure after the initialization; *weighted*, allowing the cooperative players to gain more authority in the scheme; and *verifiable* in the case of the active adversary model.

# Chapter 3

# Socio-Rational Secret Sharing

Rational secret sharing was proposed by Halpern and Teague in [41]. The authors show that, in a setting with rational players, secret sharing and multiparty computation are only possible if the actual secret reconstruction round remains unknown to the players, see Section 3.2.2. All the subsequent works use a similar approach with different assumptions. We change the direction by bridging cryptography, game theory, and reputation systems, and propose a "social model" for repeated rational secret sharing. We provide a novel scheme, named *socio-rational secret sharing* (*SRS*), in which players are invited to each game based on their reputations in the community. The players run secret sharing protocols while founding and sustaining a public trust network. As a result, new concepts such as a *rational foresighted player*, *repeated social game*, and *social Nash equilibrium* are introduced. To motivate our approach, consider a repeated secret sharing game such as "secure auctions", where the auctioneers receive sealed-bids from the bidders to compute the auction outcome without revealing the losing bids. If we assume each party has a reputation value, we can then penalize (or reward) the players who are selfish (or unselfish) from game to game. We show that this social reinforcement rationally stimulates the players to be cooperative.

## 3.1 Introduction

A new research direction was initiated by Halpern and Teague [41] in the area of secret sharing and multiparty computation in a game-theoretic setting. In this new construction, players are *rational* rather than being honest or malicious. This means each player selects his action (i.e., revealing his share or not revealing his share) based on the utility that he can gain. As illustrated by the authors, classical secret sharing fails in this setting due to the failure of the secret reconstruction round. We should highlight that, in the context of rational secret sharing, "deviation" means that a player has not revealed his share during the reconstruction phase. Sending incorrect shares is another issue which can be prevented by having the dealer sign the shares. For a simple example of such an authentication method, see [53]. We now provide a high-level description of the problem.

If players are primarily incentivized to learn the secret, and secondly, they prefer that fewer of the other parties learn it, then it is not reasonable for each player to reveal his share in the "recovery phase". For instance, suppose players $P_1, P_2, P_3$ receive shares $6, 11, 18$ from an honest dealer respectively, where $f(x) = 3 + 2x + x^2 \in \mathbb{Z}_{19}[x]$ is the secret sharing polynomial. If only two players reveal their shares in the recovery phase, then the third selfish player (who has not revealed his share) can reconstruct the secret using two revealed shares and his own private share. Obviously, the other two cooperative players who have revealed their shares do not learn the secret. This justifies why the players do not reveal their shares in a rational setting, i.e., each player waits to receive shares of the other parties (see [28, 48] for an overview in this direction).

To generalize this, consider the following scenario for a player $P_j$ where the degree of the secret sharing polynomial is $t - 1$. If $i$ players (for $i$ less than $t - 1$ or $i$ more than $t - 1$) reveal their shares, nothing changes whether $P_j$ reveals his share or not. In the former case, no one learns the secret. In the latter case, everyone learns the secret. On the other hand, if exactly $t - 1$ players reveal their shares, then $P_j$ can not only learn the secret with his own private share (i.e., $t$ shares are sufficient to use Lagrange interpolation) but also can prevent the other players from learning the secret by not revealing his share, i.e., achieving the second preference of a self-interested player in rational secret sharing. In other words, for each $P_i$, revealing the share is *weakly dominated* by not revealing the share. As a result, no one reveals his share and the secret is never reconstructed.

### 3.1.1 Motivation

In our "socio-rational" setting, the players are "selfish" and the dealer is honest, similar to standard rational secret sharing. In addition, they have "concerns" about future gain or loss since our secret sharing game is repeated an unknown number of times. We term this new type of the player, a *rational foresighted player*. In the proposed construction, each player has a reputation value that is updated according to his behavior each time the game is played. The initial reputation value is zero and its computation is public. For instance, if a player cooperates (e.g., he reveals his share), then his trust value is increased, otherwise, it is decreased. A long-term utility (used by each player for action selection) and an actual utility (used for the real payment at the end of each game) are computed based on the following parameters:

1. Estimation of future gain or loss due to trust adjustment (virtual utility).
2. Learning the secret at the current time (real utility).
3. The number of other players learning the secret at the moment (real utility).

All these factors are used by each player to estimate his long-term utility and consequently to select his action, whereas only the last two items are used to compute the real payment at the end of each game. To estimate future impact, the following scenario is considered: whenever a player cooperates (or defects), we assume he can potentially gain (or lose) some extra units of utility, i.e., he has a greater (or lesser) chance to be "invited" to the future games and consequently he gains (or loses) more utility. In other words, if the reputation of $P_i$ is decreased, he will have less chance to be invited to the future secret sharing games. Otherwise, $P_i$ is going to be invited to more secret sharing games. To realize this scenario, in each game, the dealer selects the players based on their reputations, for instance, 50% from *reputable* players, 30% from *newcomers*, and 20% from *non-reputable* parties, where the number of players in each category possibly varies. We consider the possibility of including the newcomers and non-reputable parties in order to give them a chance to participate and improve their behavior.

This gain or loss is "virtual" at the current time but will be "realized" in the future. As an example, consider the following statements, where $\mathrm{U} \gg \mathrm{u}$ and $\mathrm{V} \gg \mathrm{v}$:

1. As a consumer, if you buy something today (*cooperate* and lose \$u), you will receive a significant discount from the producer (*rewarded* \$U) on your next purchase.
2. As a producer, if you use low-grade materials to save some money (*defect* and gain \$v), you will lose many of your consumers (*penalized* \$V) in the coming years.

In other words, if we construct a socio-rational model in which the players can gain (or lose) more utility in the future games than the current game, depending on their behavior, we can then incentivize them to be foresighted and cooperative. To further motivate our concept of "socio-rational secret sharing", consider the following repeated game, as shown in Figure 3.1:

1. The bidders select a subset of auctioneers based on a probability distribution over the auctioneers' types, i.e., reputable auctioneers have a greater chance to be selected.

2. Each bidder then acts as an independent dealer to distribute the shares of his sealed-bid among the selected auctioneers.

3. Subsequently, the auctioneers compute the selling price and determine the winner by using a multiparty computation protocol.

4. In the last phase of the multiparty computation, the auctioneers reconstruct the selling price $\alpha$ and report it to the seller.

In this setting, only the auctioneers who have learned and reported $\alpha$ to the seller, are each paid \$$\Omega$, i.e., there exists a "competition" for learning the secret. In addition, \$$\Omega$ are divided among the auctioneers who have learned the secret; each of them can therefore earn more money if fewer of them learn $\alpha$. If we repeat this game an unknown number of times and choose an appropriate invitation mechanism based on the players' reputation, we can incentivize the auctioneers to be cooperative, that is, they will reveal the shares of $\alpha$ in the recovery phase.

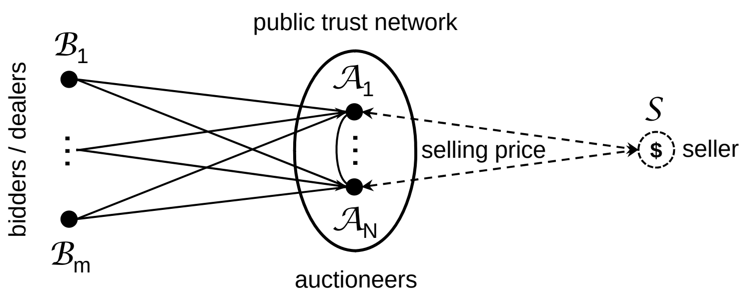

Figure 3.1: Sealed-Bid Auction as a Repeated Secret Sharing Game

### 3.1.2 Contributions

We provide a new solution concept to the rational secret sharing problem by considering a social setting in which the players enter into a long-term interaction for executing an unknown number of independent secret sharing protocols.

In our model, a public trust network is constructed to incentivize the players to be cooperative. This incentive is sustained from game to game since the players are motivated to enhance their reputations and consequently gain extra utility. In other words, they avoid a selfish behavior due to the social reinforcement of the trust network. Constructing a "social model" and inviting the players to a repeated game based on their "reputations" in the community, is a new contribution not only in rational cryptography but also in the existing game-theoretic solution concepts. We refer the reader to [60] for other discussions in this direction. Our scheme has the following desirable properties:

- It has a single recovery round, unlike the existing rational secret sharing schemes.
- It provides a unique game-theoretic solution, which is always a Nash equilibrium.
- It is immune to the rushing attack; i.e., it is not advantageous for players to wait.
- It prevents the players from aborting the game, which is a possibility in some existing solutions.

### 3.1.3 Organization

This chapter is organized as follows: Section 3.2 provides some preliminary materials. Section 3.3 reviews the literature of rational cryptography. Section 3.4 presents our construction. Section 3.5 compares our solution with the existing schemes and techniques. Finally, Section 3.6 provides concluding remarks.

## 3.2 Preliminary: Game Theory and Cryptography

In this section, some preliminary materials are provided for further technical discussions.

### 3.2.1 Game Theoretic Concepts

A *game* consists of a set of *players*, a set of *actions* and *strategies* (i.e., the way of choosing actions), and finally a *payoff function* which is used by each player to compute his utility. In *cooperative games*, players collaborate and split the total utility among themselves, i.e., cooperation is enforced by agreements. In *non-cooperative games*, players can not form agreements to coordinate their behavior, i.e., any cooperation must be self-enforcing.

The *prisoner's dilemma*, as shown in Figure 3.2, is an example of a non-cooperative game. In this game, we have two possible actions: $\mathcal{C}$: *keep quiet* (or cooperation) and $\mathcal{D}$:

*confess* (or defection). In the pay-off matrix, $+1, 0, -1$, and $-2$ denote freedom, jail for one year, jail for two years, and jail for three years respectively. The outcome of this game is going to be $(\mathcal{D}, \mathcal{D})$ due to the *Nash equilibrium* concept, while the ideal outcome is $(\mathcal{C}, \mathcal{C})$. To analyze why the game has such an outcome, consider the following two scenarios:

1. If $P_1$ selects $\mathcal{C}$ (the 1st row), then $P_2$ will select $\mathcal{D}$ (the 2nd column) since $+1 > 0$.
2. If $P_1$ selects $\mathcal{D}$ (the 2nd row), then $P_2$ will select $\mathcal{D}$ (the 2nd column) since $-1 > -2$.

This means that, regardless of whether $P_1$ cooperates or defects, $P_2$ will always defect. Since the pay-off matrix is symmetric, we also have that, regardless of whether $P_2$ cooperates or defects, player $P_1$ will always defect. In other words, since players are in two different locations and cannot coordinate their behavior, the final outcome is going to be $(\mathcal{D}, \mathcal{D})$.

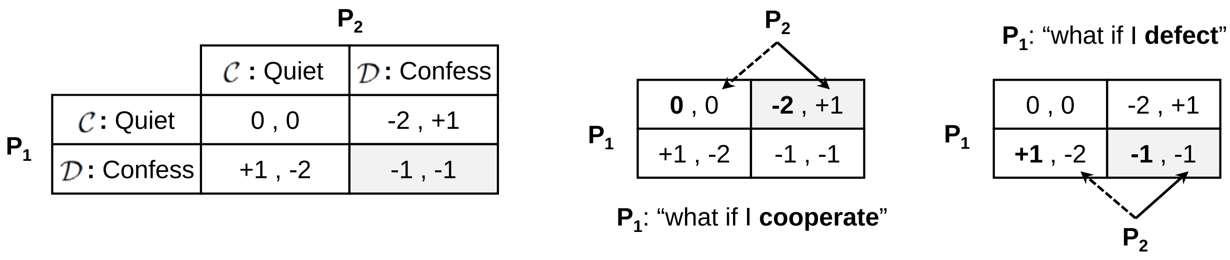

Figure 3.2: Nash Equilibrium in Prisoner's Dilemma

**Definition 3.1** *Let* $\mathcal{A} \overset{\text{def}}{=} \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ *be an action profile for n players, where* $\mathcal{A}_i$ *denotes the set of possible actions of player* $P_i$*. A* game $\Gamma = (\mathcal{A}_i, u_i)$ *for* $1 \leq i \leq n$*, consists of* $\mathcal{A}_i$ *and a utility function* $u_i : \mathcal{A} \mapsto \mathbb{R}$ *for each player* $P_i$*. We refer to a vector of actions* $\vec{a} = (a_1, \ldots, a_n) \in \mathcal{A}$ *as an* outcome of the game.

**Definition 3.2** *The* utility function $u_i$ *illustrates the preferences of player* $P_i$ *over different outcomes. We say* $P_i$ prefers *outcome* $\vec{a}$ *to* $\vec{a}'$ *iff* $u_i(\vec{a}) > u_i(\vec{a}')$*, and he* weakly prefers *outcome* $\vec{a}$ *to* $\vec{a}'$ *if* $u_i(\vec{a}) \geq u_i(\vec{a}')$*.*

In order to allow the players to follow randomized strategies (where the strategy is the way of choosing actions), we define $\sigma_i$ as a probability distribution over $\mathcal{A}_i$ for a player $P_i$. This means that he samples $a_i \in \mathcal{A}_i$ according to $\sigma_i$. A strategy is said to be a *pure strategy* if each $\sigma_i$ assigns probability 1 to a certain action, otherwise, it is said to be a *mixed strategy.* Let $\vec{\sigma} = (\sigma_1, \dots, \sigma_n)$ be the vector of players' strategies, and let $(\sigma'_i, \vec{\sigma}_{-i}) \overset{\text{def}}{=} (\sigma_1, \dots, \sigma_{i-1}, \sigma'_i, \sigma_{i+1}, \dots, \sigma_n)$, where $P_i$ replaces $\sigma_i$ by $\sigma'_i$ and all the other players' strategies remain unchanged. Therefore, $u_i(\vec{\sigma})$ denotes the expected utility of $P_i$ under the strategy vector $\vec{\sigma}$. A rational player's goal is to maximize this utility. In the following definitions, one can substitute an action $a_i \in \mathcal{A}_i$ with its probability distribution $\sigma_i \in \mathcal{S}_i$ or vice versa, where $\mathcal{S}_i$ denotes the set of possible strategies of player $P_i$.

**Definition 3.3** *A vector of strategies $\vec{\sigma}$ is a* Nash equilibrium *if, for all $i$, it holds that $u_i(\sigma'_i, \vec{\sigma}_{-i}) \leq u_i(\vec{\sigma})$. This means no one gains any advantage by deviating from the protocol as long as the other players follow the protocol.*

**Definition 3.4** *Let $\mathcal{S}_{-i} \overset{\text{def}}{=} \mathcal{S}_1 \times \cdots \times \mathcal{S}_{i-1} \times \mathcal{S}_{i+1} \times \cdots \times \mathcal{S}_n$. A strategy $\sigma_i \in \mathcal{S}_i$ (or an action) is* weakly dominated *by $\sigma'_i \in \mathcal{S}_i$ (or another action) with respect to $\mathcal{S}_{-i}$ if:*

1. *For all $\vec{\sigma}_{-i} \in \mathcal{S}_{-i}$, it holds that $u_i(\sigma_i, \vec{\sigma}_{-i}) \leq u_i(\sigma'_i, \vec{\sigma}_{-i})$.*
2. *There exists a $\vec{\sigma}_{-i} \in \mathcal{S}_{-i}$ such that $u_i(\sigma_i, \vec{\sigma}_{-i}) < u_i(\sigma'_i, \vec{\sigma}_{-i})$.*

*This means that $P_i$ can never improve its utility by playing $\sigma_i$, and he can sometimes improve it by not playing $\sigma_i$. A strategy $\sigma_i \in \mathcal{S}_i$ is* strictly dominated *if player $P_i$ can always improve its utility by not playing $\sigma_i$.*

### 3.2.2 Rational Secret Sharing

In this section, we review *rational secret sharing*, which was initiated by Halpern and Teague [41]. Their construction was later improved in [39]. The scheme consists of an honest dealer $D$, who creates a secret sharing scheme with threshold $t$, and $n$ players $P_1$, $\dots$, $P_n$.

The protocol proceeds in a sequence of iterations where only one iteration is the "real" secret recovery phase (i.e., the last iteration) and the rest are just "fake" iterations for trapping selfish players. At the end of each iteration, the protocol either terminates (due to the observation of selfish behavior or cooperation for secret recovery) or it proceeds to the next iteration. Indeed, in any given round, players do not know whether the current iteration is the real recovery phase (where a player may gain more utility by being silent and not sending his share to others), or just a test round. The following steps (a)-(d) provide a description of the initial solution to a rational secret sharing game, where $n = 3$, $t = 3$, and shares are revealed simultaneously, as presented in [41, 39]. Table 3.1 shows all the different possibilities that can occur.

(a) In each round, $D$ initiates a secret sharing scheme where each $P_i$ receives a share $f_i$.

(b) During an iteration, each $P_i$ flips a biased coin $c_i \in \{0, 1\}$ where $\mathbf{Pr}[c_i = 1] = \rho$.

(c) Players compute $c^* = \oplus c_i$ by a multiparty computation scheme without revealing $c_i$'s.

(d) Now $c^*$ is known to everyone. If $c^* = c_i = 1$, $P_i$ broadcasts his share. We then have:

(d.1) If three shares are revealed, the secret is recovered and the protocol ends.

(d.2) If $c^* = 1$, and no share or two shares are revealed, players terminate the protocol.

(d.3) In any other cases, $D$ and the players proceed to the next round, i.e., step (a).

| Rows | $c_1$ | $c_2$ | $c_3$ | Public $c^*$ | Revealed Shares |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | - |
| 2 | 0 | 0 | 1 | 1 | $f_3$ |
| 3 | 0 | 1 | 0 | 1 | $f_2$ |
| 4 | 0 | 1 | 1 | 0 | - |
| 5 | 1 | 0 | 0 | 1 | $f_1$ |
| 6 | 1 | 0 | 1 | 0 | - |
| 7 | 1 | 1 | 0 | 0 | - |
| 8 | 1 | 1 | 1 | 1 | $f_1, f_2, f_3$ |

Table 3.1: Three-Player Rational Secret Sharing Game

To see how the above protocol works, assume $P_1, P_2$ follow the protocol whereas $P_3$ is willing to deviate. He may deviate in "coin-tossing" or in "revealing" his share. We should note that each $P_i$ selects $c_i$ independently. The following cases are different possible deviation scenarios:

- It is not advantageous for $P_3$ to bias $c_3$ to be 0 with higher probability, since, when $c_3 = 0$, either no share or one share is revealed.
- It is also not advantageous for $P_3$ to bias $c_3$ to be 1 with higher probability, since, when $c_3 = 1$, either no share, or one share, or all shares are revealed. This may lead to an "early" secret recovery but it does not have any effect of the utility of $P_3$.

- If $c_3 = 0$ or $c^* = 0$ (that is, one of rows $1, 3, 4, 5, 6, 7$ in Table 3.1 occurs), then there is no incentive for player $P_3$ to deviate since in all these cases he is supposed not to reveal his share.
- If $c_3 = 1$ and $c^* = 1$ (that is, one of rows $2, 8$ in Table 3.1 occurs), then player $P_3$ is supposed to reveal his share. There exist two possibilities:

  1. $c_1 = 1$ and $c_2 = 1$, which occurs with the following probability:

$$\begin{aligned}\mathbf{Pr}[c_1 = 1 \wedge c_2 = 1 | c_3 = 1 \wedge c^* = 1] &= \frac{\mathbf{Pr}[c_1 = 1 \wedge c_2 = 1 \wedge c_3 = 1]}{\mathbf{Pr}[c_3 = 1 \wedge c^* = 1]} \\ &= \frac{\rho^3}{(1-\rho)(1-\rho)\rho + \rho^3} \\ &= \frac{\rho^2}{(1-\rho)^2 + \rho^2}\,.\end{aligned}$$

  2. $c_1 = 0$ and $c_2 = 0$, which occurs with the remaining probability:

$$\begin{aligned}\mathbf{Pr}[c_1 = 0 \wedge c_2 = 0 | c_3 = 1 \wedge c^* = 1] &= \frac{\mathbf{Pr}[c_1 = 0 \wedge c_2 = 0 \wedge c_3 = 1]}{\mathbf{Pr}[c_3 = 1 \wedge c^* = 1]} \\ &= \frac{(1-\rho)(1-\rho)\rho}{(1-\rho)(1-\rho)\rho + \rho^3} \\ &= \frac{(1-\rho)^2}{(1-\rho)^2 + \rho^2}\,.\end{aligned}$$

Therefore, if player $P_3$ deviates by not revealing his share, either he is going to be the only player who learns the secret or the protocol terminates and he never learns the secret. Let assume player $P_3$ gains $U^+$ if he is the only player who learns the secret, let $U$ denotes the utility gain for each $P_i$ if all three players learn the secret, and let $U^-$ denotes the utility gain, say \$0, for each player $P_i$ if no one learns the secret. It is assumed that $U^+ > U > U^-$. Therefore, a rational $P_3$ will cheat only if:

$$U^+\left(\frac{\rho^2}{(1-\rho)^2 + \rho^2}\right) + U^-\left(\frac{(1-\rho)^2}{(1-\rho)^2 + \rho^2}\right) > U. \qquad (3.1)$$

If we assign an appropriate value to $\rho$, based on the players' utility function, such that the inequality (3.1) is not satisfied, then player $P_3$ has no incentive to deviate when $c_3 = 1$ and $c^* = 1$.

The authors in [41] also showed that this three-player game can be generalized to a game with $n$ players. As we just stated, certain assumptions regarding the players' utility function are required for rational secret sharing to be achievable. Let $u_i(\vec{a})$ denote the utility of $P_i$ in a specific outcome $\vec{a}$ of the protocol. Suppose $l_i(\vec{a})$ is a bit defining whether $P_i$ has learned the secret or not in $\vec{a}$. We then define $\delta(\vec{a}) = \sum_i l_i(\vec{a})$, which denotes the number of players who have learned the secret. The generalized assumptions of rational secret sharing are as follows:

- $l_i(\vec{a}) > l_i(\vec{a}') \Rightarrow u_i(\vec{a}) > u_i(\vec{a}')$.
- $l_i(\vec{a}) = l_i(\vec{a}')$ and $\delta(\vec{a}) < \delta(\vec{a}') \Rightarrow u_i(\vec{a}) > u_i(\vec{a}')$.

The first assumption means $P_i$ prefers an outcome in which he learns the secret, that is, since $l_i(\vec{a}) = 1$ and $l_i(\vec{a}') = 0$, he therefore prefers $\vec{a}$. The second assumption means $P_i$ prefers an outcome in which the fewest number of other players learn the secret, given that $P_i$ learns (or does not learn) the secret in both outcomes.

## 3.3 Previous Works: Rational Secret Sharing

As we mentioned, the notion of *rational secret sharing* was introduced by Halpern and Teague [41], and it was later improved in [39]. Assuming the same game-theoretic model, Lysyanskaya and Triandopoulos [57] provide a solutions in a *mixed behavior* setting in which players are either rational or malicious. Abraham et al. [2] define a notion of resistance to coalitions and present a *coalition-resistant* protocol. All these constructions use simultaneous channels (either a broadcast channel or secure private channels) that means each player must decide on the value he wants to broadcast before observing the values broadcasted by the other players; this is known as a *strategic game.*

The proposed protocols in [54, 55, 45] rely on *physical assumptions* such as secure envelopes and ballot boxes, which might be impossible or difficult to implement. In the same model, Micali and shelat [64] provided a purely rational secret sharing scheme using a verifiable trusted channel. They showed that all the existing solutions not only rely on the players' rationality, but also on their beliefs. As a result, they cannot guarantee that all rational players learn the secret. For instance, suppose $P_i$ believes that equilibrium $(a, b)$ is played whereas $P_j$ believes $(a', b')$ is played, but the game leads to $(a, b')$, which may not be an equilibrium at all.

Kol and Naor [53] introduced an equilibrium notion, termed *strict Nash equilibrium*, in an information-theoretic secure setting. In a Nash equilibrium, no deviations are advantageous (i.e., there is no incentive to deviate). In its strict counterpart, all deviations are disadvantageous (i.e., there is an incentive not to deviate). They first considered both simultaneous and non-simultaneous broadcast channels and provided a new solution to avoid the simultaneous channel at the cost of increasing the round complexity.

Kol and Naor later [52] showed that all the existing computational-based protocols are susceptible to backward induction because of the cryptographic primitives used in the beginning of those protocols. That is, they can surely be broken after an exponential number of rounds. The authors then illustrate a new cryptographic coalition-resilient approach that is *immune to backward induction* by considering simultaneous as well as non-simultaneous broadcast channels.

The notion of *computational strict Nash equilibrium* was introduced in [32]. This construction is dealer-free and can tolerate a coalition of size $t-1$ without using simultaneous channels. It can even be run over asynchronous point-to-point networks. Finally, it is efficient in terms of computation, share size, and round complexity.

Maleka et al. [61] presented *repeated rational secret sharing*, with the same approach proposed in [79], by considering two punishment strategies. In the first strategy, each player reveals his share as long as the other players cooperate. As soon as the first defection is observed, the players do not reveal their shares in every subsequent game. In the second strategy, the players do not send their shares to the deviant for $k$ subsequent games after observing the first defection. In the first scheme, each player not only punishes the deviant but also the other players including himself. In the second method, a player may deviate in an *expensive* secret recovery without having any concern for $k$ subsequent *cheap* reconstructions. Indeed, the nature of a punishment strategy must depend on how much future outcomes are worth for each player. Finally, they only considered a fixed number of $m$ players without allowing newcomers to join the scheme.

Other results have recently been proposed in the literature. For instance, Ong et al. [78] presented a protocol that is fair when the reconstruction phase is executed with many rational players together with a minority of honest parties. Asharov and Lindell [4] explained that in all the existing protocols, the designer needs to know the actual utility values of the players. They then showed that it is possible to achieve utility independence through the relaxation of assumptions. Gradwohl et al. [40] provided the definitions of computational solution concepts that guarantee sequential rationality. Finally, Asharov et al. [3] demonstrated how game theoretic concepts can be used to capture cryptographic notions of security.

## 3.4 Socio-Rational Secret Sharing

We first provide formal definitions of a *social game*, a *social Nash equilibrium*, and *socio-rational secret sharing*. In our model, each $P_i$ has a public reputation value $\mathcal{T}_i$, where $\mathcal{T}_i(0) = 0$ and $-1 \le \mathcal{T}_i(p) \le +1$; $p = 0, 1, 2, \ldots$ denote the time periods of the games. The construction of this function is independent of our protocol; therefore, we use the existing function presented in Section 1.5. Since the importance of each game might be different, it would be possible to consider the *transaction cost* parameter (as stated in Section 1.5) during trust adjustment, but we do not use this in the thesis. We assume each player's action $a_i \in \{\mathcal{C}, \mathcal{D}, \perp\}$, where $\mathcal{C}$ and $\mathcal{D}$ denote "cooperation" and "defection" respectively, and $\perp$ denotes $P_i$ has not been chosen by the dealer to participate in the current game.

**Definition 3.5** *In a society of size $N$, a* social game $\Gamma = (\mathcal{A}_i, \mathcal{T}_i, u_i, u'_i)$, *where $1 \le i \le N$, is repeatedly played an unbounded number of times among different subsets of players. Each $P_i$ has a set of* actions $\mathcal{A}_i$, *a* reputation value $\mathcal{T}_i$, *a* long-term utility function $u_i$, *and an* actual utility function $u'_i$. *Let $\mathcal{A} \overset{\text{def}}{=} \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$ be the action profile. In each game:*

- *A subset of $n \le N$ players is chosen by the dealer for each new secret sharing game based on their reputation values $\mathcal{T}_i$, where more reputable players have a greater chance to be selected.*
- *Each $P_i$ estimates his long-term utility by $u_i : \mathcal{A} \times \mathcal{T}_i \mapsto \mathbb{R}$ based on his gain in the current game and future games. Player $P_i$ then selects his action $a_i$ according to $u_i$.*
- *Let $\vec{a} = (a_1, \ldots, a_N) \in \mathcal{A}$ be the current game's outcome. The actual utility of each $P_i$ is computed based on a function $u'_i : \mathcal{A} \mapsto \mathbb{R}$ at the end of the current game.*
- *Each player's reputation value $\mathcal{T}_i$ is publicly updated by a trust function based on each player's action in the current game, as shown in Section 1.5, except that $\mathcal{T}_i(p) = \mathcal{T}_i(p-1)$ if $a_i = \perp$.*

Note that the *long-term utility function $u_i$* is used for "action selection" and the *actual utility function $u'_i$* is used to compute the "real gain" at the end of the current game.

**Definition 3.6** *A vector of strategies $\vec{\sigma}$ is said to be a* social Nash equilibrium *in each game of a social game $\Gamma$ if for all $i$ and any $\sigma'_i \neq \sigma_i$ it holds that $u_i(\sigma'_i, \vec{\sigma}_{-i}) \le u_i(\vec{\sigma})$. Accordingly, if $u_i(\sigma'_i, \vec{\sigma}_{-i}) < u_i(\vec{\sigma})$, it is said to be a* strict social Nash equilibrium. *That is, considering future games, a player cannot gain any benefit by deviating from the protocol in the current game.*

In the next sections, we discuss the utility function as a central component in every game. This is due to the fact that players make decisions based on this function. Note that the *utility assumption* refers to the players' preferences over the game's outcome whereas the *utility computation* shows the method of computing the utility of each player.

### 3.4.1 Utility Assumption

Let $u_i(\vec{a})$ denotes $P_i$'s utility resulting from a list of players' actions $\vec{a}$ by considering future games, let $u'_i(\vec{a})$ denotes $P_i$'s utility resulting from the current game, let $l_i(\vec{a}) \in \{0, 1\}$ denote if $P_i$ has learned the secret during a given time period, and define $\delta(\vec{a}) = \sum_i l_i(\vec{a})$. Finally, let $\mathcal{T}_i^{\vec{a}}(p)$ denote the reputation of $P_i$ after outcome $\vec{a}$ in period $p$; each game of a social game is played in a single period. The generalized assumptions of socio-rational secret sharing are as follows:

*A.* $l_i(\vec{a}) = l_i(\vec{a}')$ and $\mathcal{T}_i^{\vec{a}}(p) > \mathcal{T}_i^{\vec{a}'}(p) \Rightarrow u_i(\vec{a}) > u_i(\vec{a}')$.

*B.* $l_i(\vec{a}) > l_i(\vec{a}') \Rightarrow u'_i(\vec{a}) > u'_i(\vec{a}')$.

*C.* $l_i(\vec{a}) = l_i(\vec{a}')$ and $\delta(\vec{a}) < \delta(\vec{a}') \Rightarrow u'_i(\vec{a}) > u'_i(\vec{a}')$.

The preference "*A*" illustrates that, whether player $P_i$ learns the secret or not, $P_i$ prefers to maintain a high reputation. The preferences "*B*" and "*C*" are the standard assumptions of rational secret sharing.

**Definition 3.7** *In a social game, a* rational foresighted player *has prioritized assumptions: "A" (greediness) is strictly preferred to "B" and has an impact factor $\rho_1$, "B" (selfishness) is at least as good as "C" and has an impact factor $\rho_2$, and "C" (selfishness) has an impact factor $\rho_3$. We denote this using the notation $A^{\rho_1} \succ B^{\rho_2} \succeq C^{\rho_3}$, where $\rho_1 > \rho_2 \geq \rho_3 \geq 1$.*

The above definition reflects the fact that a rational foresighted player has a "long-term" vision and firstly prefers to achieve the highest level of trustworthiness. Only in this case, he will be involved in the future games and consequently gain more profits (interpreted as greediness). He secondly prefers an outcome in which he learns the secret. Finally, he desires the fewest number of other players learn the secret. We next propose a long-term utility function that satisfies all three preferences.

### 3.4.2 Utility Computation

Our long-term utility function $u_i : \mathcal{A} \times \mathcal{T}_i \mapsto \mathbb{R}$ computes the utility that each player $P_i$ potentially gains or loses by considering future games, based on assumptions "$A$", "$B$", "$C$", whereas the actual utility function $u'_i : \mathcal{A} \mapsto \mathbb{R}$ only computes the current gain or loss in a given time period, based on assumptions "$B$" and "$C$".

**Sample Function.**

We define two functions $\omega_i(\vec{a})$ and $\tau_i(\vec{a})$ for the $n$ participating players of the current game:

$$\omega_i(\vec{a}) = \frac{3}{2 - \mathcal{T}_i^{\vec{a}}(p)} \tag{3.2}$$

$$\tau_i(\vec{a}) = \mathcal{T}_i^{\vec{a}}(p) - \mathcal{T}_i^{\vec{a}}(p-1). \tag{3.3}$$

Since $-1 \leq \mathcal{T}_i^{\vec{a}}(p) \leq +1$, then $+1 \leq \omega_i(\vec{a}) \leq +3$. Let $\Omega > 0$ be a "unit of utility", for instance, \$100. To satisfy our assumptions in Section 3.4.1, we define:

$$A \;:\; \frac{|\tau_i(\vec{a})|}{\tau_i(\vec{a})} \times \omega_i(\vec{a}) \times \Omega \quad \text{where} \quad \frac{|\tau_i(\vec{a})|}{\tau_i(\vec{a})} = \begin{cases} +1 \text{ if } a_i = \mathcal{C} \\ -1 \text{ if } a_i = \mathcal{D} \end{cases} \tag{3.4}$$

$$B \;:\; l_i(\vec{a}) \times \Omega \quad \text{where} \quad l_i(\vec{a}) \in \{0, 1\} \tag{3.5}$$

$$C \;:\; \frac{l_i(\vec{a})}{\delta(\vec{a}) + 1} \times \Omega \quad \text{where} \quad \delta(\vec{a}) = \sum_{i=1}^{N} l_i(\vec{a}). \tag{3.6}$$

- (3.4) will evaluate to $+\omega_i(\vec{a})\Omega$ if $P_i$ cooperates and it will evaluate to $-\omega_i(\vec{a})\Omega$, otherwise. This means that $P_i$ gains or loses at least $1\Omega$ and at most $3\Omega$ (depending on his reputation value, as reflected in $\omega_i$) units of utility in the future games due to his current behavior.
- (3.5) illustrates that a player gains one unit of utility if he learns the secret in the current game and he loses this opportunity, otherwise.
- (3.6) results in "almost" one unit of utility being divided among all the players $P_i$ who have learned the secret in the current game; to avoid a division by 0 when $\delta(\vec{a}) = 0$, we use $\delta(\vec{a}) + 1$ in the denominator.

We combine these three terms, weighted with their corresponding impact factors, as follows:

$$u'_i(\vec{a}) \quad = \quad \rho_2\Big(l_i(\vec{a}) \times \Omega\Big) + \rho_3\Big(\frac{l_i(\vec{a})}{\delta(\vec{a})+1} \times \Omega\Big), \tag{3.7}$$

and then we have:

$$\begin{aligned} u_i(\vec{a}) \quad &= \quad \rho_1\Big(\frac{|\tau_i(\vec{a})|}{\tau_i(\vec{a})} \times \omega_i(\vec{a}) \times \Omega\Big) + u'_i(\vec{a}) \\ &= \quad \Omega \times \Bigg(\rho_1\Big(\frac{|\tau_i(\vec{a})|}{\tau_i(\vec{a})} \times \omega_i(\vec{a})\Big) + \rho_2\Big(l_i(\vec{a})\Big) + \rho_3\Big(\frac{l_i(\vec{a})}{\delta(\vec{a})+1}\Big)\Bigg). \end{aligned} \tag{3.8}$$

The function $u_i(\vec{a})$ shows that if $P_i$, with preference factors $\rho_1 > \rho_2 \geq \rho_3 \geq 1$, defects (or cooperates), he may gain (or lose) $\rho_2\Omega + (\rho_3\Omega)/(\delta(\vec{a})+1)$ utility in the current game, but he will lose (or gain) "$x$" units of utility in the future games, where $\rho_1\Omega \leq x \leq 3\rho_1\Omega$. That is, future loss or gain is more important than the current loss or gain. We later show that the dealer gives a lesser (or a greater) chance of contribution to non-reputable (or reputable) players in the future games; i.e., reputation remains with a player as a characteristic which continuously affects his utility.

We will prove that in our socio-rational secret sharing scheme, cooperation is a strict Nash equilibrium. Our proofs are based on the sample function described above. However, the proofs can be generalized to apply to any utility function that satisfies the requirements of Definition 3.7.

### 3.4.3 Proposed Protocol

We now discuss our socio-rational secret sharing scheme; the details are presented in Figure 3.3. Suppose the public trust network has already been created. Assume we have a dealer who initiates a $(t, n)$-threshold secret sharing scheme. Also, as in previous research on rational secret sharing, we assume all the players use a "pure strategy". A *socio-rational secret sharing* game $\Gamma = (\mathcal{A}_i, \mathcal{T}_i, u_i, u'_i)$ is a social game that is played among rational foresighted players and it is based on the following elements:

1. Set of possible actions $\mathcal{A}_i = \{\mathcal{C}, \mathcal{D}, \perp\}$, as defined in Section 3.4.

2. Trust function $\mathcal{T}_i$, except that $\mathcal{T}_i(p) = \mathcal{T}_i(p-1)$ if $a_i = \perp$, as defined in Section 1.5.

3. Long-term utility function $u_i : \mathcal{A} \times \mathcal{T}_i \mapsto \mathbb{R}$, as defined in Section 3.4.2.

4. Actual utility function $u'_i : \mathcal{A} \mapsto \mathbb{R}$, as defined in Section 3.4.2.

Secret Sharing

1. Let $\phi$ be the current probability distribution over players' types $\mathcal{B}, \mathcal{N}, \mathcal{G}$, as defined in Section 1.5. The dealer $D$ first selects $n$ out of $N$ players, where $n \leq N$, based on this non-uniform probability distribution.

2. The dealer $D$ then initiates a $(t, n)$-secret sharing scheme by selecting $f(x) \in \mathbb{Z}_q[x]$ of degree $t-1$, where $f(0) = \alpha$ is the secret. Subsequently, he sends shares $f(i)$ to $P_i$ for the $n$ chosen players, and leaves the scheme.

Secret Recovery

1. Each chosen player $P_i$ computes his long-term utility function $u_i : \mathcal{A} \times \mathcal{T}_i \mapsto \mathbb{R}$, and then selects an action, i.e., revealing or not revealing his share $f(i)$.

2. If enough shares are revealed, the polynomial $f(x)$ is reconstructed through Lagrange interpolation and the secret $f(0) = \alpha$ is recovered.

3. Each chosen player $P_i$ receives his utility $u'_i : \mathcal{A} \mapsto \mathbb{R}$ (i.e., the real payment) at the end of the reconstruction phase according to the outcome.

4. Finally, the reputation values $\mathcal{T}_i$ of all the chosen players are publicly updated according to each player's behavior and the trust function.

Figure 3.3: Socio-Rational Secret Sharing Protocol

The *sharing phase* is similar to that of standard secret sharing. The only difference is the way that the dealer selects $n$ out of $N$ players for secret sharing. In other words, the dealer gives more chance to reputable players compared to unreliable parties. Although a natural approach is to invite only the reputable players, it might be preferable if the dealer provides opportunities for newcomers as well as the bad players. Once in a while, he could give a chance to the bad players so they can compensate for their past behavior. This is a realistic approach even in human society; it can be interpreted as a "forgiveness factor". The *secret recovery phase* is also similar to that of the standard secret sharing but with some extra components.

We should mention that the players' reputations and the trust function are public information. Therefore, all computations associated with the reputation system can be done by any authority or a committee of the players. It is also worth mentioning that it is not required to consider unknown number of iterations for secret recovery, which is the case in all the existing rational secret sharing schemes. In fact, in a "socio-rational secret sharing" game, we have an unknown number of independent secret sharing games, whereas in "rational secret sharing", we only have one secret with an unknown number iterations for secret recovery.

**Theorem 3.8** *In a* $(2,2)$*-socio-rational secret sharing,* $\mathcal{C}$ *strictly dominates* $\mathcal{D}$*, considering a long-term utility function, shown in Equation (3.8), which satisfies the preferences of rational foresighted players, shown in Definition 3.7. In other words,* $\mathcal{D}$ *is strictly dominated by* $\mathcal{C}$*. As a result,* $(\mathcal{C},\mathcal{C})$ *is a strict social Nash equilibrium that is a unique solution.*

**Proof.** We compute the utility of each outcome for player $P_i$. Let $P_j$ be the other player.

1. If both players cooperate, denoted by $(\mathcal{C},\mathcal{C})$, then $\tau_i$ is positive, $l_i = 1$ since $P_i$ has learned the secret, and $\delta = 2$ because both players have learned the secret. We have:

$$\big(\tau_i > 0, l_i = 1, \delta = 2\big) \Rightarrow u_i^{(\mathcal{C},\mathcal{C})}(\vec{a}) = \Omega\Big(\rho_1\omega_i + \rho_2 + \frac{\rho_3}{3}\Big).$$

2. If only $P_i$ cooperates, denoted by $(\mathcal{C},\mathcal{D})$, then $\tau_i$ is positive, $l_i = 0$ since $P_i$ has not learned the secret, and $\delta = 1$ because only player $P_j$ has learned the secret. We have:

$$\big(\tau_i > 0, l_i = 0, \delta = 1\big) \Rightarrow u_i^{(\mathcal{C},\mathcal{D})}(\vec{a}) = \Omega\Big(\rho_1\omega_i\Big).$$

3. If only $P_j$ cooperates, denoted by $(\mathcal{D},\mathcal{C})$, then $\tau_i$ is negative, $l_i = 1$ since $P_i$ has learned the secret, and $\delta = 1$ because only player $P_i$ has learned the secret. We have:

$$\big(\tau_i < 0, l_i = 1, \delta = 1\big) \Rightarrow u_i^{(\mathcal{D},\mathcal{C})}(\vec{a}) = \Omega\Big(-\rho_1\omega_i + \rho_2 + \frac{\rho_3}{2}\Big).$$

4. If both players defect, denoted by $(\mathcal{D},\mathcal{D})$, then $\tau_i$ is negative, $l_i = 0$ since $P_i$ has not learned the secret, and $\delta = 0$ because no one has learned the secret. We have:

$$\big(\tau_i < 0, l_i = 0, \delta = 0\big) \Rightarrow u_i^{(\mathcal{D},\mathcal{D})}(\vec{a}) = \Omega\Big( - \rho_1\omega_i\Big).$$

We can ignore the common factor $\Omega$. We know $1 \leq \omega_i(\vec{a}) \leq 3$ and $\rho_1 > \rho_2 \geq \rho_3 \geq 1$.

- First, we have:

$$u_i^{(\mathcal{C},\mathcal{C})}(\vec{a}) = \rho_1\omega_i + \rho_2 + \frac{\rho_3}{3} > \rho_1\omega_i = u_i^{(\mathcal{C},\mathcal{D})}(\vec{a}). \tag{3.9}$$

- Next, it is easy to see that

$$u_i^{(\mathcal{C},\mathcal{D})}(\vec{a}) = \rho_1\omega_i > -\rho_1\omega_i + \rho_2 + \frac{\rho_3}{2} = u_i^{(\mathcal{D},\mathcal{C})}(\vec{a}) \tag{3.10}$$

  if and only if $2\rho_1\omega_i > \rho_2 + \dfrac{\rho_3}{2}$. We have:

$$\begin{aligned} 2\rho_1\omega_i &\geq 2\rho_1 \\ &> \rho_2 + \rho_3 \\ &> \rho_2 + \frac{\rho_3}{2}, \end{aligned}$$

  so the desired conclusion follows.

- Finally,

$$u_i^{(\mathcal{D},\mathcal{C})}(\vec{a}) = -\rho_1\omega_i + \rho_2 + \frac{\rho_3}{2} > -\rho_1\omega_i = u_i^{(\mathcal{D},\mathcal{D})}(\vec{a}). \tag{3.11}$$

Consequently, we have the following payoff inequalities which proves the theorem:

$$\overbrace{u_i^{(\mathcal{C},\mathcal{C})}(\vec{a}) > u_i^{(\mathcal{C},\mathcal{D})}(\vec{a})}^{P_i \text{ cooperates}} > \overbrace{u_i^{(\mathcal{D},\mathcal{C})}(\vec{a}) > u_i^{(\mathcal{D},\mathcal{D})}(\vec{a})}^{P_i \text{ defects}}.$$

∎

The interesting observation is the difference between the two utilities $u_i^{(\mathcal{C},\mathcal{D})}(\vec{a})$ and $u_i^{(\mathcal{D},\mathcal{C})}(\vec{a})$. This means that it is better for player $P_i$ to cooperate, even though he might not learn the secret and the other party might learn it. On the other hand, even if $P_i$ learns

| $P_1 \backslash P_2$ | $\mathcal{C}$ooperation | $\mathcal{D}$efection |
|---|---|---|
| $\mathcal{C}$ooperation | $\mathcal{U}$,$\mathcal{U}$ | $\mathcal{U}^{--}$,$\mathcal{U}^{+}$ |
| $\mathcal{D}$efection | $\mathcal{U}^{+}$,$\mathcal{U}^{--}$ | $\mathcal{U}^{-}$,$\mathcal{U}^{-}$ |

Table 3.2: $(2, 2)$-SS with Selfish Players

| $P_1 \backslash P_2$ | $\mathcal{C}$ooperation | $\mathcal{D}$efection |
|---|---|---|
| $\mathcal{C}$ooperation | $\mathcal{U}^{+}$,$\mathcal{U}^{+}$ | $\mathcal{U}$,$\mathcal{U}^{-}$ |
| $\mathcal{D}$efection | $\mathcal{U}^{-}$,$\mathcal{U}$ | $\mathcal{U}^{--}$,$\mathcal{U}^{--}$ |

Table 3.3: $(2, 2)$-Socio-Rational SS

the secret by deviating at a given period (using the share of the other party), he will gain less utility in the long-term. This is due to future gain or loss and the significance of being reputable, which is incorporated in our long-term utility function by considering an impact factor $\rho_1$. We should also note that, as $\rho_1$ is increased, the difference between $u_i^{(\mathcal{C},\mathcal{D})}(\vec{a})$ and $u_i^{(\mathcal{D},\mathcal{C})}(\vec{a})$ also increases, i.e., the enforcement for cooperation would be greater.

In a secret sharing scheme with selfish players, the outcome ($\mathcal{U}^{-}$,$\mathcal{U}^{-}$) is a Nash equilibrium, as shown in Table 3.2, where $\mathcal{U}^{+} > \mathcal{U} > \mathcal{U}^{-} > \mathcal{U}^{--}$. Rational secret sharing solves this problem by using a randomized mechanism, as presented in Section 3.2.2. The payoff matrix associated with socio-rational secret sharing is illustrated in Table 3.3. In this payoff matrix, the outcome ($\mathcal{U}^{+}$,$\mathcal{U}^{+}$) is a *strict social Nash equilibrium.*

We should note that our socio-rational game is a non-cooperative game. In fact, cooperation is self-enforcing due to the importance of reputation as well as future concerns of a rational foresighted player. In a cooperative game, this enforcement is provided by a third party and players do not really compete. Moreover, this payoff matrix does not mean that the players never deviate. As an example, consider a scenario in which a player is involved in many independent social games. If he simultaneously receives many requests for secret recovery of various schemes, he will select the one in which he can gain more utility. This is discussed later, in Section 3.4.4.

We now analyze our socio-rational secret sharing in a setting where $n > 2$ players take part in each secret sharing game.

**Theorem 3.9** *In a socio-rational secret sharing scheme with $n$ participants and $t = 2$, $\mathcal{C}$ strictly dominates $\mathcal{D}$ for all players $P_i$, assuming the preferences of rational foresighted parties. Consequently, the vector $\vec{a}^{\mathcal{C}} = (a_1^{\mathcal{C}}, \ldots, a_n^{\mathcal{C}})$ is a strict social Nash equilibrium that is a unique solution.*

**Proof.** Let $\mathcal{C}_i$ (or $\mathcal{D}_i$ ) denote that player $P_i$ cooperates (or defects), and let $\mathcal{C}_{-i}$ (or $\mathcal{D}_{-i}$) denote that, excluding $P_i$, all the other players cooperate (or defect), and finally let $\mathcal{M}_{-i}$

denotes that, excluding $P_i$, some players cooperate and some of them defect, that is, we have both $\mathcal{C}$ooperation and $\mathcal{D}$efection. We compute the utility of each outcome based on Equation (3.8) for the least possible threshold $t = 2$ when $n > 2$, that is, when two shares are enough to learn any secret.

1. If all the players cooperate, denoted by $(\mathcal{C}_i, \mathcal{C}_{-i})$, then $\tau_i$ is positive, $l_i = 1$ since player $P_i$ has learned the secret, and $\delta = n$ because all the players have learned the secret. We have:

$$\big(\tau_i > 0, l_i = 1, \delta = n\big) \Rightarrow u_i^{(\mathcal{C}_i,\mathcal{C}_{-i})}(\vec{a}) = \Omega\Big(\rho_1\omega_i + \rho_2 + \frac{\rho_3}{n+1}\Big).$$

2. If player $P_i$ cooperates but some of the other parties cooperate and some defect, denoted by $(\mathcal{C}_i, \mathcal{M}_{-i})$, then $\tau_i$ is positive, $l_i = 1$, and $\delta = n$ because all the players have learned the secret. We have:

$$\big(\tau_i > 0, l_i = 1, \delta = n\big) \Rightarrow u_i^{(\mathcal{C}_i,\mathcal{M}_{-i})}(\vec{a}) = \Omega\Big(\rho_1\omega_i + \rho_2 + \frac{\rho_3}{n+1}\Big).$$

3. If only $P_i$ cooperates, denoted by $(\mathcal{C}_i, \mathcal{D}_{-i})$, then $\tau_i$ is positive, $l_i = 0$, and $\delta = n - 1$ because all the players, except $P_i$, have learned the secret. We have:

$$\big(\tau_i > 0, l_i = 0, \delta = n-1\big) \Rightarrow u_i^{(\mathcal{C}_i,\mathcal{D}_{-i})}(\vec{a}) = \Omega\Big(\rho_1\omega_i\Big).$$

4. If only $P_i$ defects, denoted by $(\mathcal{D}_i, \mathcal{C}_{-i})$, then $\tau_i$ is negative, $l_i = 1$, and $\delta = n$ because all the players have learned the secret. We have:

$$\big(\tau_i < 0, l_i = 1, \delta = n\big) \Rightarrow u_i^{(\mathcal{D}_i,\mathcal{C}_{-i})}(\vec{a}) = \Omega\Big(-\rho_1\omega_i + \rho_2 + \frac{\rho_3}{n+1}\Big).$$

5. If player $P_i$ defects but some of the other parties cooperate and some defect, denoted by $(\mathcal{D}_i, \mathcal{M}_{-i})$, then $\tau_i$ is negative, $l_i = 1$, and $\delta = n - 1$ if only one player reveals his share, or $\delta = n$ if at least two players reveal their shares. We have:

$$\big(\tau_i < 0, l_i = 1, \delta\big) \Rightarrow u_i^{(\mathcal{D}_i,\mathcal{M}_{-i})}(\vec{a}) = \Omega\Big(-\rho_1\omega_i + \rho_2 + \frac{\rho_3}{\delta+1}\Big),$$

   where $\delta \in \{n-1, n\}$.

6. If all the players defect, denoted by $(\mathcal{D}_i, \mathcal{D}_{-i})$, then $\tau_i$ is negative, $l_i = 0$, and $\delta = 0$ because no one has learned the secret. We have:
$$\Big(\tau_i < 0, l_i = 0, \delta = 0\Big) \Rightarrow u_i^{(\mathcal{D}_i, \mathcal{D}_{-i})}(\vec{a}) = \Omega\Big(-\rho_1\omega_i\Big).$$

We now analyze these six scenarios:

- If player $P_i$ cooperates (cases 1–3), regardless of whether the other players cooperate or defect, then
$$u_i^{\mathcal{C}}(\vec{a}) \geq \rho_1\omega_i. \tag{3.12}$$
- If player $P_i$ defects (cases 4–6), regardless of whether the other players cooperate or defect, then
$$u_i^{\mathcal{D}}(\vec{a}) \leq -\rho_1\omega_i + \rho_2 + \frac{\rho_3}{n}. \tag{3.13}$$

It is easy to prove that $\rho_1\omega_i > -\rho_1\omega_i + \rho_2 + \frac{\rho_3}{n}$. In fact, the proof is essentially the same as the proof of (3.10) in Theorem 3.8. Therefore, it is always in $P_i$'s best interest to cooperate:
$$u_i^{\mathcal{C}}(\vec{a}) > u_i^{\mathcal{D}}(\vec{a}).$$

■

**Remark 3.10** *A similar analysis can be given for $t > 2$ with any number of players.*

### 3.4.4 Expected Utility

In this section, we illustrate how each $P_i$ can compute his expected utility when he participates in different independent social games, whereas in the previous section, we studied a player's participation in a single social game. Note that the *utility value* shows the connection between actions and their corresponding consequences for a player, whereas the *expected utility* of $P_i$ is an estimation of gain or loss when he plays with another player $P_j$.

We initially show how to compute the expected utilities in a $(2, 2)$-game for "cooperation" and "defection". An expected utility is computed as a linear combination of utility values and the probability of $P_j$'s cooperation, where $\epsilon_j \in [0, 1]$ denotes the probability that the opponent $P_j$ may cooperate and $\mathcal{U}^+ > \mathcal{U} > \mathcal{U}^- > \mathcal{U}^{--}$ are the utility values from Table 3.3. We have:

$$\begin{aligned}\mathcal{EU}_i^{\mathcal{C}}(\vec{a}) &= \epsilon_j\,\mathcal{U}^{+} + (1-\epsilon_j)\,\mathcal{U} && (3.14)\\ \mathcal{EU}_i^{\mathcal{D}}(\vec{a}) &= \epsilon_j\,\mathcal{U}^{-} + (1-\epsilon_j)\,\mathcal{U}^{--} && (3.15)\end{aligned}$$

**Theorem 3.11** *In a socio-rational secret sharing game with two players $P_i$ and $P_j$, the expected utility of cooperation is greater than the expected utility of defection, that is, $\mathcal{EU}_i^{\mathcal{C}}(\vec{a}) - \mathcal{EU}_i^{\mathcal{D}}(\vec{a}) > 0$, where $\epsilon_j$ is the probability of opponent's cooperation.*

**Proof.**

$$\begin{aligned}\mathcal{EU}_i^{\mathcal{C}}(\vec{a}) - \mathcal{EU}_i^{\mathcal{D}}(\vec{a}) &= \big(\epsilon_j\,\mathcal{U}^{+} + (1-\epsilon_j)\,\mathcal{U}\big) - \big(\epsilon_j\,\mathcal{U}^{-} + (1-\epsilon_j)\,\mathcal{U}^{--}\big) \quad \text{by (3.14) (3.15)}\\ &= \epsilon_j\,(\mathcal{U}^{+} - \mathcal{U}^{-}) + (1-\epsilon_j)\,(\mathcal{U} - \mathcal{U}^{--})\\ &> 0.\end{aligned}$$

∎

We now consider the expected utilities in two independent $(2,2)$-games. Let us define $\mathcal{EU}_i^{\mathcal{C}}(\vec{a}_{ij})$ and $\mathcal{EU}_i^{\mathcal{C}}(\vec{a}_{ik})$ as the expected utilities of the two games, when player $P_i$ cooperates with players $P_j$ and $P_k$ respectively.

**Theorem 3.12** *Suppose $P_i$ plays with $P_j$ and $P_k$ in two independent $(2,2)$-games. Player $P_i$ then gains more utility if he collaborates with the most reputable player.*

**Proof.** Assume $P_j$ and $P_k$ have different reputation values computed with the same trust function. We make the plausible assumption that $\epsilon_j > \epsilon_k$ if $P_j$ is more reputable than $P_k$. Suppose $P_i$ receives the same unit of utility $\Omega$ in both games, and let $\vec{a}_{ij}, \vec{a}_{ik}$ be the outcomes of the two games respectively. We therefore have:

$$\begin{aligned}\mathcal{EU}_i^{\mathcal{C}}(\vec{a}_{ij}) - \mathcal{EU}_i^{\mathcal{C}}(\vec{a}_{ik}) &= \big(\epsilon_j\,\mathcal{U}^{+} + (1-\epsilon_j)\,\mathcal{U}\big) - \big(\epsilon_k\,\mathcal{U}^{+} + (1-\epsilon_k)\,\mathcal{U}\big) \quad \text{by (3.14)}\\ &= \epsilon_j\,\mathcal{U}^{+} - \epsilon_k\,\mathcal{U}^{+} + (1-\epsilon_j)\,\mathcal{U} - (1-\epsilon_k)\,\mathcal{U}\\ &= (\epsilon_j - \epsilon_k)\,\mathcal{U}^{+} + (\epsilon_j - \epsilon_k)\,\mathcal{U}\\ &> 0,\end{aligned}$$

since $\epsilon_j > \epsilon_k$. As a result, $\mathcal{EU}_i^{\mathcal{C}}(\vec{a}_{ij}) > \mathcal{EU}_i^{\mathcal{C}}(\vec{a}_{ik})$. This means that player $P_i$ gains more utility if he collaborates with $P_j$ rather than $P_k$. ∎

## 3.5 Comparison with Existing Techniques

Our contribution differs from *rational secret sharing* and *social secret sharing*, as shown in Figure 3.4. Our scheme is a repeated game that addresses the problem of secret recovery in the presence of rational foresighted parties, whereas:

- "rational secret sharing" is a one-time game with repeated rounds, and it deals with the problem of secret recovery of a secret in the presence of rational players, and
- "social secret sharing" defines how many shares each player can hold in a weighted secret sharing scheme with honest and malicious parties.

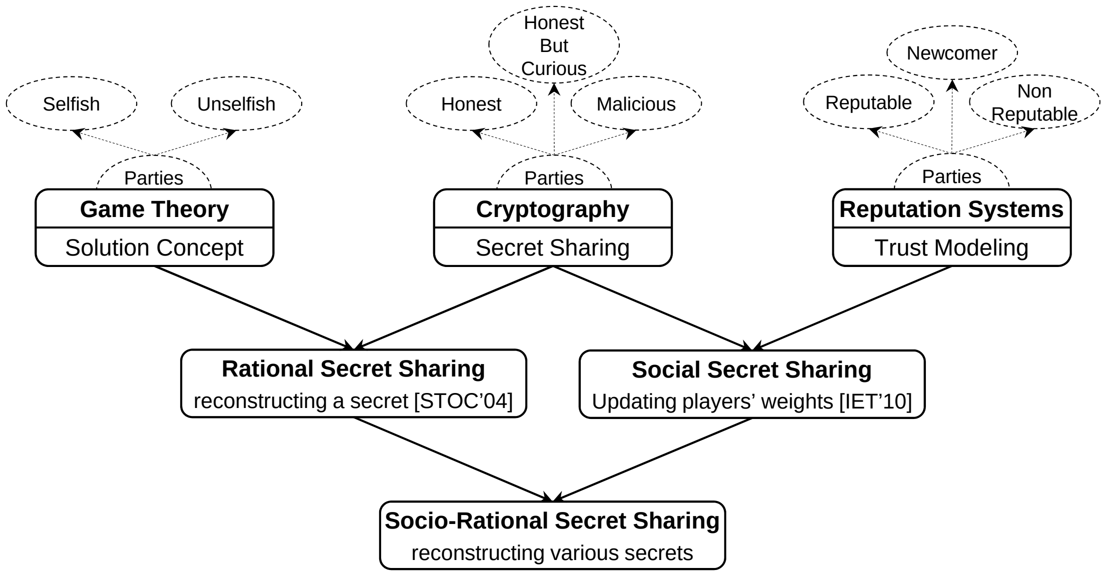

Figure 3.4: Pedigree of Socio-Rational Secret Sharing

Our contribution is also different from the *punishment strategy* used in the *repeated prisoners' dilemma* [79] where the players penalize potential deviants. As the authors have mentioned, the major point behind the repeated games is the fact that if each participant believes any deviation terminates the mutual cooperation (resulting in a subsequent loss that outweighs the short-term gain), he then prefers to cooperate. For instance, consider the prisoners' dilemma with $\mathcal{C}$*ooperation* and $\mathcal{D}$*efection* actions. Both players cooperate until one of them deviates. Then, the other player chooses $\mathcal{D}$ for a specific number of times as a punishment. Meanwhile, the deviant rewards the punisher by selecting $\mathcal{C}$ as a compensation. Finally, the game returns to a mutual cooperation. Our approach has the following advantages over the punishment strategy:

- In our model, a player is not just an abstract entity who selects actions. He also has a social characteristic reflected in his reputation that shows his trustworthiness. This attribute is solely determined by the player's actions.
- The punishment strategy is performed by selecting actions that are harmful for deviants whereas, in our model, punishment or reward (losing or gaining reputation and utility) is independent of action selection.
- Our approach avoids penalizing innocent players or the punisher himself. It also avoids being involved, to some extent, in a game with seriously selfish players who are not reputable (due to our "invitation approach").
- The punishment strategy does not consider that a game may have various levels of importance and utility weights when it is repeatedly played. For instance, whether it is a secret sharing scheme to launch a "missile" or to open a "safety box".
- The punishment strategy has a discrete penalizing approach whereas our construction has a continuous impact on the deviants. For example, it may take a long time for a player to regain lost reputation.
- Our proposed approach not only considers punishment and reward but also defines six different scenarios in order to fairly deal with various types of players, including good players, bad players, and newcomers.

Our contribution is also different from the constructions forming histories and beliefs such as *subgame perfect equilibrium* or *Bayesian equilibrium* [79]. In the former, players reassess their decisions based on the past *history*; i.e., a sequence of previous actions. In the latter, the game is designed to deal with the situations in which parties are not certain about the characteristics of each other. Therefore, they form *beliefs*; i.e., probability distributions over actions, to anticipate any future behavior.

Let $P_i$ be a specific player, and let $P_j$ for $1 \leq j \neq i \leq n$ denote any other player except $P_i$. Our trust calculation method and social setting differs from these kinds of solution concepts in the following aspects:

- In forming a belief about $P_i$'s intentions, both parties contribute. That is, $P_i$ is indirectly involved by his behavior, i.e., action selections, and the other players are directly involved by the methodology that they use in order to form the probability distribution over actions. A belief may or may not be common knowledge, meaning that various players may have different judgments and beliefs about $P_i$. On the other hand, the reputation of $P_i$ in a trust network is solely determined by his behavior through a trust function, which is a commonly known function for reputation measurement. That is, the reputation is a direct reflection of $P_i$'s attitude (there is no misunderstanding), and he knows the impact of his decision on the other players (i.e., whether he is known as a good player, a bad player, or a newcomer). He can also estimate how much extra utility he may gain or lose after his reputation's adjustment, which is a strong enforcement.

- Histories and beliefs are more general compared to the reputation system in a trust network. This means a belief as a probability distribution can be defined over any set of actions for any types of players. On the other hand, reputation is built over a specific set of actions, such as $\mathcal{C}$ooperation and $\mathcal{D}$efection ($\mathcal{X}$: corruption as a malicious behavior might be also considered in a mixed model), for specific types of players, such as good players, bad players, and newcomers. As a result, the reputation system is simpler and it is more suitable for cryptographic constructions.

- In the history and belief systems, all the measurements are "inside" the game-theoretic model whereas our reputation system isolates these computations from the game. For instance, two separate probability distributions can be defined over the players' types and actions by considering their past behavior[1]. But our publicly known trust function combines these two measurements in a single reputation value outside of the game-theoretic model (although these values might be interpreted similar to "types" and "beliefs"). In other words, the punishment or reward is embedded inside of our reputation system which continuously affects the players' utilities in the game-theoretic model; i.e., losing utility due to the reputation's decline or losing reputation and not being selected in the future secret sharing games.

[1]For instance, suppose that $P_i$ is good or bad, as defined in Section 1.5, with probabilities 0.7 or 0.3 respectively. Based on these values, a $P_j$ may believe that $P_i$ reveals or denies to reveal his share with certain probabilities, e.g., 0.9 or 0.1 respectively.

## 3.6 Conclusion

This chapter provides a multidisciplinary construction that connects three major areas of computer science in order to propose a new solution for one of the most fundamental cryptographic primitives.

As we stated, the social network with reputation consideration is a strong enforcement for the participants to cooperate, for instance, a player may change his non-cooperative approach after analyzing his reputation, or after estimating his future loss. In our social setting, bad players can compensate for their past behavior by continuous cooperation. On the other hand, reputable players can gain more profits as long as they act properly, and newcomers can fairly start their social interactions.

Finally, we should note that having a trust network by considering long-term interactions can be seen as a new direction in game theory itself, specifically, the theoretical models used in social sciences such as economics and political science because elements in those frameworks are more close to human social behavior.

# Chapter 4

# Dynamic Secret Sharing

In this chapter, we first provide a comprehensive analysis of existing *dynamic secret sharing* (*DSS*) schemes in both the passive and active adversary models. We then provide several new techniques where the threshold and/or the secret can be changed multiple times to arbitrary values after the initialization. Finally, we introduce a new application of dynamic threshold schemes, named *sequential secret sharing* (*SQS*), in which several secrets with increasing thresholds are shared among the players who have different levels of authority.

## 4.1 Introduction

In a threshold scheme, the sensitivity of the secret as well as the number of players may fluctuate due to various reasons, for instance, mutual trust may vary or the structure of the players' organization might be changed. A possible solution to this problem is to modify the threshold and/or change the secret. Moreover, a common problem with almost all secret sharing schemes is that they are "one-time", meaning that the secret and shares are known to everyone after secret recovery. This problem could be resolved if the dealer shares various secrets at the beginning, but a better solution is to dynamically generate new secrets in the absence of the dealer. These issues are our main motivation to revisit dynamic threshold schemes.

In the literature, a "dynamic scheme" refers to a scheme with threshold and/or access structure changeability. To the best of our knowledge, the existing protocols update the threshold without changing the secret. It is also worth mentioning that secret changeability can be used in other cryptographic constructions, as we later show in Section 4.5.1.

The goal of *threshold changeability* is to convert a $(t, n)$-threshold scheme into a $(t', n)$-threshold scheme, where we allow both $t < t'$ (*threshold increase*) and $t' < t$ (*threshold reduction*). We should note that, in the case of threshold increase, if the decision is to keep the same secret (a desirable property in the case where the secret is difficult or expensive to change), it is clear that at least $n - t + 1$ players have to erase their old shares honestly (otherwise, the secret can be constructed by an unauthorized set from "old" shares). It has been previously noted that erasing old shares is an inevitable assumption in a threshold changeable scheme with a constant secret [62], as well as in proactive secret sharing schemes [80, 44]. Otherwise, the secret must be changed (this issue is discussed in Section 4.5). Here, we assume all the players erase their old shares when updating shares.

### 4.1.1 Motivation

We are motivated to revisit dynamic threshold schemes due to various issues. As stated by Martin et al. [62], in a threshold scheme, the sensitivity of the secret as well as the number of players may fluctuate due to various reasons. For instance,

(a) mutual trust might be decreased over time, perhaps because of organizational problems or security incidents, and

(b) the structure of the organization to which the players belong might be changed, for example, new players may join the organization and need to be incorporated into the security structure, or current parties may leave the organization while still retaining a copy of their shares.

In both of these cases, the threshold should be adjusted accordingly. In other words, modifying the threshold and/or changing the secret might be required throughout the lifetime of a secret. This requirement is more significant when the lifetime is increased. In the literature, there exist some techniques to address threshold changeability but existing solutions suffer from one or more of the following drawbacks:

- They assume the existence of an online trusted authority.
- They have a large storage requirement because it is necessary to predistribute extra shares relating to different thresholds.
- They are limited to predefined modifications of the thresholds.

Another well-known problem with many secret sharing schemes is that they are one-time, meaning that, after recovering the secret in a public computation, the secret and the shares are known to everyone. To resolve this problem, the concept of *multistage secret sharing* is proposed [43], where many secrets are shared by the dealer ahead of time. However, a better solution is to generate new secrets in the absence of the dealer.

### 4.1.2 Contributions

We review previous literature on dynamic threshold schemes in Section 4.2. In later sections, we provide various techniques for dynamic secret sharing that are more general and simpler as compared to existing solutions. Indeed, our proposed methods have various desirable properties. We assume the existence of a dealer who initializes the scheme. However, the players can change the secret and the threshold after the scheme's initialization without the participation of the dealer. We note that it is possible to make the scheme completely dealer-free if the initialization is performed by the players themselves using a secure MPC protocol, but we do not follow this approach in this thesis.

Our protocols are *unconditionally secure* in that they do not rely on any computational assumptions. They have *minimum storage cost* since players do not need to store extra shares beforehand to modify the parameters of the scheme. Finally, they are *flexible* because the secret and the threshold can be changed to arbitrary values multiple times.

In Section 4.3, we consider the passive adversary setting. First, we discuss a commonly-used "re-sharing technique", also known as 2-level sharing, that modifies the threshold to any arbitrary value. This approach applies the Lagrange method in order to combine the players' shares after re-sharing. As an alternative solution, we show that this linear combination can equivalently be carried out by using a Vandermonde matrix, as was done in [36] in order to reduce the threshold to a desired value after performing a multiplication of secrets. Finally, we modify a folklore method to reduce the threshold based on revealing one or more shares. In the new variation we propose, the players first construct and then reveal an "extra" share, and then they use this extra share to modify their existing shares so that the threshold is decreased but the secret remains unchanged. We call this new technique *public evaluation*.

In Section 4.4, we turn to the active adversary model, where the problem is more difficult. We recall a re-sharing scheme discussed in [67] to provide two observations with respect to this approach. This method fails to increase the threshold, and moreover, it is not secure against a mobile adversary (a fact that was already noted in [67] without a formal proof). Therefore, we provide two separate solutions to achieve threshold modification:

(a) To decrease the threshold, we first extend the public evaluation method from the previous section to the active adversary setting.

(b) To increase the threshold, we add shares of a higher-degree polynomial $g$ (having a zero constant term and random coefficients of all non-constant terms) to the players' shares on the original secret sharing polynomial $f$. As a result, the threshold is increased. To generate this polynomial $g$, we develop a protocol that we term *polynomial production*, which allows a subset of players to jointly create a random polynomial with an unknown secret in the active adversary setting. This polynomial can then be used to generate the desired polynomial $g$.

In Section 4.5, we use the polynomial production protocol in order to change both the secret and the threshold. We also motivate our approach by presenting a new application of dynamic threshold schemes, named *sequential secret sharing*. In a sequential secret sharing scheme, players progressively construct a sequence of secret sharing schemes with different (but related) secrets and thresholds in the absence of the dealer.

### 4.1.3 Organization

This chapter is organized as follows. Section 4.2 reviews dynamic threshold schemes. Section 4.3 and Section 4.4 analyze threshold changeability in the passive and active adversary models respectively. Section 4.5 illustrates how the threshold and the secret can be changed simultaneously. Section 4.6 provides concluding remarks.

## 4.2 Previous Works: Dynamic Threshold Schemes

Martin et al. [62] designed a threshold changeable secret sharing scheme, in the absence of secure channels, based on two methods. The first one was implemented by the Shamir approach and the second one is a geometric construction. They made two assumptions. First, the original shares must contain the required information for extracting both the shares of the initial scheme and the shares of the future scheme, known as *shares* and *subshares* respectively. As a result, the size of the stored shares grows linearly with the number of required modifications to the threshold. Second, the proposed construction assumes that shareholders behave honestly in the sense that they only use the subshares that are relevant to the threshold in current use.

Using the previous approach, Maeda et al. [59] proposed an unconditionally secure verifiable scheme where the threshold can be changed several times, say $N$ times, but only to values that are determined in advance. In this protocol, each player receives one full share and extracts the subsequent subshares from it by applying a sequence of $N$ public functions released by the dealer during initialization. The dealer also has to distribute $N$ polynomials ahead of time. The authors assume that the secret is not recovered before the threshold changes, and therefore no shares have been pooled before secret reconstruction.

Steinfeld et al. [96] constructed a dynamic threshold scheme for Shamir secret sharing. The general idea is that players add an appropriate amount of random noise to their shares in order to create subshares that contain incomplete information regarding the primary shares. As a result, $t$ subshares are not sufficient to recover the secret, but by using a larger number of subshares, say $t'$ where $t' > t$, the secret can be reconstructed.

Tartary and Wang [102] proposed a dealer-free threshold changeable scheme in which the problem of secret recovery is reduced to the polynomial reconstruction problem. In this construction, players send some fake shares along with their real shares to increase the threshold $t$ at the side of the combiner to a new value $t'$. First, the threshold stays constant among players. Second, their algorithm does not allow any value $t'$ to be chosen; i.e., it must be predefined.

In addition to the drawbacks we mentioned regarding the existing techniques, there is one common problem with all of these solutions. That is, if an adversary attacks the shareholders (not the combiner) then he can have access to the original shares, shares related to various thresholds, and/or shares without any noise. In any of these situations, the secret can be recovered by the attacker.

Other methods have been proposed in the literature to achieve threshold changeability in a secret sharing scheme, for instance:

- re-sharing existing shares of a $(t, n)$-threshold scheme by a set of new polynomials of degree $t'$ [30];
- redistribution of secret shares to new access structures in which participants of a scheme send information to a new set of players in such a way that the old secret is shared among a new access structure [27, 63]; and
- dynamic secret sharing schemes, where the dealer triggers a specific access structure out of a given set, or enables the players to recover various secrets in different times by sending them the same broadcast message [14].

The paper [6] also considers schemes with changeable parameters, e.g., the threshold and the number of players, in order to minimize the storage costs (size of shares) and the size of broadcast messages. Finally, we assume that all players erase their old shares after computing the new ones.

## 4.3 Schemes in the Passive Adversary Model

First, we discuss a re-sharing technique that modifies the threshold to any arbitrary value using a technique based on Lagrange interpolation. We then show that re-sharing can equivalently be done by using a Vandermonde matrix, as was done in [36]. Finally, we present our method of public share evaluation, where we modify a folklore technique to reduce the threshold based on revealing one or more shares. The players first construct and then reveal an "extra" share, and then they use this extra share to modify their existing shares so that the threshold is decreased but the secret remains unchanged.

In the passive adversary model, there are only secure private channels between each pair of players. All computations are done in a finite field $\mathbb{Z}_q$ where $q$ is a prime number.

### 4.3.1 Threshold Modification by the Lagrange Method

The idea of re-sharing shares in secret sharing has been used in many papers through the years. One of the first instances of this technique is found in [30] in the context of threshold RSA. The general idea is to re-share the existing shares of a $(t, n)$-threshold scheme by a set of polynomials of degree $t'$. We note that this method of threshold modification is described informally in [67, Section 5].

Suppose an honest dealer initiates a Shamir secret sharing scheme and then leaves. That is, he randomly generates $f(x) \in \mathbb{Z}_q[x]$ of degree at most $t-1$ in which its constant term is the secret $f(0) = \alpha$, and then sends share $f(i)$ to player $P_i$ for $1 \leq i \leq n$. Then each player re-shares his share using a new random polynomial of degree at most $t'-1$. The detailed description of the scheme is given in Figure 4.1. We also present an example of this scheme after describing the protocol.

**Theorem 4.1** *The threshold modification protocol presented in Figure 4.1 is secure under the passive adversary model tolerating $t-1$ colluders, and correctly computes the secret $\alpha$.*

Threshold Modification

1. Each player $P_i$ selects a random polynomial $g_i(x)$ of degree at most $t' - 1$ such that $g_i(0) = f(i)$. He then gives $g_i(j)$ to $P_j$ for $1 \leq j \leq n$, i.e., re-sharing the original shares by auxiliary shares. The share-exchange matrix $\mathcal{E}_{n\times n}$, where each player generates a row and receives a column, is as follows:

$$\mathcal{E}_{n\times n} = \begin{pmatrix} g_1(1) & g_1(2) & \dots & g_1(n) \\ g_2(1) & g_2(2) & \dots & g_2(n) \\ \vdots & \vdots & \ddots & \vdots \\ g_n(1) & g_n(2) & \dots & g_n(n) \end{pmatrix} \text{ where } g_i(0) = f(i). \tag{4.1}$$

2. At this step, a set $\Delta$ is determined such that it consists of the identifiers of at least $t$ elected players. Then, the following public constants are computed:

$$\gamma_i^{\Delta} = \prod_{j\in\Delta, j\neq i} \frac{j}{j-i} \quad \text{where } 1 \leq i, j \leq n \text{ represent players' } ids. \tag{4.2}$$

3. Each player $P_j$ erases his old shares, and then combines the auxiliary shares he has received from other players to compute his new share as follows:

$$\varphi_j = \sum_{i\in\Delta} \left(\gamma_i^{\Delta} \times g_i(j)\right). \tag{4.3}$$

Secret Recovery

- Now, if a set $\Delta'$ of at least $t'$ players $P_j$ cooperate, they can recover $\alpha$ by using the Lagrange interpolation method:

$$\alpha = \sum_{j\in\Delta'} \left(\gamma_j^{\Delta'} \times \varphi_j\right). \tag{4.4}$$

Figure 4.1: Threshold Modification by the Lagrange Method in the Passive Adv. Model

**Proof.** It is clear that a set $\nabla$ of colluders, where $|\nabla| = t-1$, is not able to recover the secret after the initial distribution of shares by the dealer. In the next stage, the players re-share their shares using polynomials $g_i(x)$ of degree $t'-1$. As a result, $t'-1$ colluders cannot reconstruct any of these re-sharing polynomials in order to reveal the shares of the good players, because $t'-1$ players have no information about any of $t$ original shares. Since all players erase their old shares after computing the new ones, there is no way to construct the secret by making use of the original shares. Therefore, the protocol is secure under the passive (honest-but-curious) adversary model.

Now, we show the correctness of the scheme (where $|\Delta| \geq t$ and $|\Delta'| \geq t'$):

$$\begin{aligned}
\sum_{j\in\Delta'}\left(\gamma_j^{\Delta'} \times \varphi_j\right) &= \sum_{j\in\Delta'}\left(\gamma_j^{\Delta'} \times \sum_{i\in\Delta}\left(\gamma_i^{\Delta} \times g_i(j)\right)\right) && \text{by (4.3)}\\
&= \sum_{i\in\Delta}\left(\gamma_i^{\Delta} \times \sum_{j\in\Delta'}\left(\gamma_j^{\Delta'} \times g_i(j)\right)\right) && \\
&= \sum_{i\in\Delta}\left(\gamma_i^{\Delta} \times g_i(0)\right) && \text{by (1.3)}\\
&= \sum_{i\in\Delta}\left(\gamma_i^{\Delta} \times f(i)\right) && \text{by (4.1)}\\
&= f(0) = \alpha && \text{by (1.3).}
\end{aligned}$$

■

**Example 4.2** *The dealer first distributes shares of $f(x) = 3+2x+x^2 \in \mathbb{Z}_{19}$, where $t=3$, among four players: $f(1)=6, f(2)=11, f(3)=18, f(4)=8$.*

1. *Suppose the players re-share their shares with new polynomials of degree 3, i.e., $t'=4$:*

$$\begin{aligned}
f_1(x) &= \boldsymbol{6} + x + x^2 + 2x^3 & \qquad f_3(x) &= \boldsymbol{18} + 3x + 2x^2 + x^3\\
f_2(x) &= \boldsymbol{11} + 2x + x^2 + 3x^3 & \qquad f_4(x) &= \boldsymbol{8} + 2x + 2x^2 + 2x^3
\end{aligned}$$

*Matrix $\mathcal{E}_{4\times4}$, where each $P_i$ generates a row and receives a column, is as follows:*

$$\mathcal{E}_{4\times4} = \begin{pmatrix} 10 & 9 & 15 & 2\\ 17 & 5 & 12 & 18\\ 5 & 2 & 15 & 12\\ 14 & 17 & 10 & 5 \end{pmatrix}.$$

2. *Players need to define a set* $\Delta$ *of size* 3 *in order to update their shares. Suppose* $\Delta = \{P_1, P_2, P_3\}$*; then we have*

$$\gamma_1 = \frac{(0-2)(0-3)}{(1-2)(1-3)} = 3 \quad \gamma_2 = \frac{(0-1)(0-3)}{(2-1)(2-3)} = -3 \quad \gamma_3 = \frac{(0-1)(0-2)}{(3-1)(3-2)} = 1$$

3. *Players update their shares based on the exchanged shares, and erase their old shares:*

$$\begin{aligned} \varphi_1 &= (3)10 + (-3)17 + (1)5 = -16 & \varphi_3 &= (3)15 + (-3)12 + (1)15 = 5 \\ \varphi_2 &= (3)9 + (-3)5 + (1)2 = 14 & \varphi_4 &= (3)2 + (-3)18 + (1)12 = -17 \end{aligned}$$

*The secret* $\alpha$ *can be reconstructed by the set* $\Delta' = \{P_1, P_2, P_3, P_4\}$ *of size* 4 *as follows:*

$$\begin{aligned} \alpha &= \frac{(0-2)(0-3)(0-4)}{(1-2)(1-3)(1-4)}(-16) + \frac{(0-1)(0-3)(0-4)}{(2-1)(2-3)(2-4)}(14) \\ &\quad + \frac{(0-1)(0-2)(0-4)}{(3-1)(3-2)(3-4)}(5) + \frac{(0-1)(0-2)(0-3)}{(4-1)(4-2)(4-3)}(-17) \\ &\equiv -16 \bmod 19 \\ &= 3. \end{aligned}$$

### 4.3.2 Threshold Modification by a Vandermonde Matrix

In this section, we provide an alternative method for threshold modification where the secret remains the same. The idea is to re-share the existing shares of the players and then use a Vandermonde matrix to change the threshold from $t$ to $t'$. This approach was initially proposed in [36] (which is a simplified version of [9]) for "degree reduction". We present the protocol in Figure 4.2 and we then provide an example of this scheme.

We are not going to discuss this protocol in detail. Mainly we just want to point out that this approach is basically equivalent to the protocol from the previous subsection that uses the Lagrange method. (This equivalence is rather obvious, but it does not seem to be explicitly stated in the existing literature.) The only difference, which is mainly a matter of presentation, is that the Lagrange method uses the shares belonging to any set of at least $t$ players to compute the updated shares in the scheme, while the Vandermonde method uses the shares of all $n$ players.

Threshold Modification

1. The re-sharing phase is similar to the first step of the previous protocol, which was presented in Figure 4.1.

2. Participants then compute the first row of a publicly known matrix $\mathcal{V}^{-1}_{n\times n}$ (mod $q$) to adjust the threshold, where $\mathcal{V}_{n\times n}$ is the Vandermonde matrix, i.e., $\mathcal{V}_{i,j} = i^{(j-1)}$ for $1 \le i, j \le n$. Suppose this vector is $\mathcal{V}^{-1}_{1\times n} = (v_1 \quad v_2 \quad \dots \quad v_n)$.

3. Eventually, each player $P_j$ computes his final share by multiplying $\mathcal{V}^{-1}_{1\times n}$ by his vector of shares:

$$\varphi(j) = \begin{pmatrix} v_1 & v_2 & \dots & v_n \end{pmatrix} \begin{pmatrix} g_1(j) \\ g_2(j) \\ \vdots \\ g_n(j) \end{pmatrix} = \sum_{i=1}^{n} v_i \, g_i(j).$$

Secret Recovery

- To recover the secret, $t'$ participants $P_j$ have to collaborate in order to construct a polynomial of degree $t'-1$:

$$\varphi(x) = \sum_{j=1}^{t'} \left( \prod_{1\le i\le t', i\ne j} \frac{x-i}{j-i} \times \varphi(j) \right).$$

  Players then compute secret $\varphi(0)$. Alternatively, $\varphi(0)$ can be reconstructed directly using the Lagrange Interpolation formula, without first computing $\varphi(x)$.

Figure 4.2: Threshold Modification by a Vandermonde Matrix in the Passive Adv. Model

**Example 4.3** *Consider the polynomial $f(x)$ from Section 4.2 with the same four players. Since the first step is exactly the same as the first phase of the Lagrange method, we start from the second step as follows:*

2. *Players compute the first row of a publicly known matrix $\mathcal{V}^{-1}_{n\times n}$ (mod $q$) to adjust the threshold, where $\mathcal{V}_{n\times n}$ is the Vandermonde matrix:*

$$\mathcal{V}_{n\times n} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 8 \\ 1 & 4 & 16 & 7 \end{pmatrix} \qquad \mathcal{V}^{-1}_{n\times n} = \begin{pmatrix} \mathbf{4} & \mathbf{13} & \mathbf{4} & \mathbf{18} \\ 2 & 0 & 12 & 5 \\ 11 & 15 & 13 & 18 \\ 3 & 10 & 9 & 16 \end{pmatrix}$$

3. *Eventually, each player $P_j$ computes his final share by multiplying $\mathcal{V}^{-1}_{1\times n}$ by his vector of exchanged shares:*

$$\varphi(1) = \begin{pmatrix} 4 & 13 & 4 & 18 \end{pmatrix} \begin{pmatrix} 10 \\ 17 \\ 5 \\ 14 \end{pmatrix} = 1 \qquad \varphi(3) = \begin{pmatrix} 4 & 13 & 4 & 18 \end{pmatrix} \begin{pmatrix} 15 \\ 12 \\ 15 \\ 10 \end{pmatrix} = 0$$

$$\varphi(2) = \begin{pmatrix} 4 & 13 & 4 & 18 \end{pmatrix} \begin{pmatrix} 9 \\ 5 \\ 2 \\ 17 \end{pmatrix} = 16 \qquad \varphi(4) = \begin{pmatrix} 4 & 13 & 4 & 18 \end{pmatrix} \begin{pmatrix} 2 \\ 18 \\ 12 \\ 5 \end{pmatrix} = 0$$

*The secret $\alpha$ can be reconstructed as follows, where the new threshold $t' = 4$:*

$$\begin{aligned} \alpha &= \frac{(0-2)(0-3)(0-4)}{(1-2)(1-3)(1-4)}(1) + \frac{(0-1)(0-3)(0-4)}{(2-1)(2-3)(2-4)}(16) \\ &\quad + \frac{(0-1)(0-2)(0-4)}{(3-1)(3-2)(3-4)}(0) + \frac{(0-1)(0-2)(0-3)}{(4-1)(4-2)(4-3)}(0) \\ &\equiv 212 \bmod 19 \\ &= 3. \end{aligned}$$

### 4.3.3 Threshold Decrease by Public Evaluation

As we mentioned previously, a folklore method to reduce the threshold in a secret sharing scheme consists of revealing one or more shares. However, there are some issues that make this approach inconvenient:

- If a player reveals his share, then he effectively removes himself as a player in the secret sharing scheme.
- A dealer can easily construct and reveal a “new” share to the players, but this requires an online dealer.
- In any event, the remaining players have to store the revealed share as well as their old share.

Our approach enables the players to first jointly construct and then reveal an “extra” share, and then use this extra share to modify their existing shares so that the threshold is decreased but the secret remains unchanged. The players can use the *enrollment protocol* from Section 2.5 to publicly generate a new share at a new point different from the players' existing *ids*. Each player then combines the revealed share with his own private share in order to decrease the threshold. For instance, suppose $P_1, P_2, P_3$ have three shares at points $x = 1, 2, 3$ (respectively) on a polynomial $f$ of degree 2. If they jointly construct and reveal a share at a new point, say $x = 4$, they can then combine this share with their own private shares such that the degree of $f$ is decreased to 1 but the secret remains unchanged; we shortly explain how to perform this combination.

Suppose $f(x) \in \mathbb{Z}_q[x]$ of degree $t-1$ is the secret sharing polynomial. Let $\Gamma \subseteq \mathbb{Z}_q \backslash \{0\}$ denote the set of players' *ids*. As a result, each $P_i$ for $i \in \Gamma$ receives the share $f(i) \in \mathbb{Z}_q$ from the dealer. For the sake of simplicity, suppose the players want to decrease the threshold from $t$ to $t-1$ in the absence of the dealer (for further threshold reduction, they can just repeat the same procedure again). The proposed protocol is presented in Figure 4.3 (the steps (1–4) are the same as Figure 2.5). An example of the scheme is given subsequently.

**Theorem 4.4** *The threshold reduction approach presented in Figure 4.3 is secure under the passive adversary model tolerating $t-1$ colluders, and it correctly reduces the threshold.*

**Proof.** The security of the first four steps is provided in Theorem 2.4. We just need to show that the new shares are on a polynomial $\hat{f}(x)$ of degree at most $t-2$ where $f(0) = \hat{f}(0)$. Assuming that the new share $(j, f(j))$ has been revealed, we have:

$$(x - j) \mid \big(f(x) - f(j)\big).$$

Let us define $f^*(x)$ of degree $t-2$ as follows:

Threshold Decrease

1. The players select an *id* $j$ such that $j \notin \mathcal{P}$. Subsequently, $t$ players $P_i$ are selected (e.g., $1 \leq i \leq t$). They compute Lagrange constants as follows:

$$\gamma_i = \prod_{1 \leq k \leq t, i \neq k} \frac{j-k}{i-k}.$$

2. Each $P_i$ multiplies his share $f(i)$ by his Lagrange constant. He then randomly splits the result into $t$ portions, that is, $f(i) \times \gamma_i = \partial_{1i} + \partial_{2i} + \cdots + \partial_{ti}$ for $1 \leq i \leq t$.

3. The players exchange $\partial_{ki}$'s through pairwise channels (in a fashion similar to Figure 4.1). As a result, each player $P_k$ holds $t$ values. He adds them together and reveals $\sigma_k = \sum_{i=1}^{t} \partial_{ki}$ to everyone.

4. The players add these values $\sigma_k$ for $1 \leq k \leq t$ together to compute the public share $f(j) = \sum_{k=1}^{t} \sigma_k$.

5. Each $P_i$ combines his private share $f(i)$ with the public share $f(j)$ as follows:

$$\hat{f}(i) = f(j) - j\left(\frac{f(i) - f(j)}{i - j}\right). \tag{4.5}$$

6. The shares $\hat{f}(i)$ are on a new polynomial $\hat{f}(x) \in \mathbb{Z}_q[x]$ of degree at most $t-2$ where $\hat{f}(0) = f(0)$. Therefore, $t-1$ players are now sufficient to recover the secret.

Figure 4.3: Threshold Decrease by Public Evaluation in the Passive Adversary Model

$$f^*(x) \stackrel{\text{def}}{=} \frac{f(x) - f(j)}{x - j}. \tag{4.6}$$

Using (4.6), the share of each player $P_i$ on $f^*$ is obtained as follows:

$$f^*(i) = \frac{f(i) - f(j)}{i - j}. \tag{4.7}$$

Accordingly, using (4.6), the secret associated with $f^*$ will be:

$$f^*(0) = \frac{f(0) - f(j)}{-j}. \tag{4.8}$$

Using (4.8), the relation between the old secret $f(0)$ and the new secret $f^*(0)$ is obtained:

$$f(0) = f(j) - jf^*(0). \tag{4.9}$$

Suppose we define the new polynomial

$$\hat{f}(x) = f(j) - jf^*(x) = f(j) - j\left(\frac{f(x) - f(j)}{x - j}\right). \tag{4.10}$$

Then $\hat{f}(x)$ has degree at most $t - 2$ and $\hat{f}(0) = f(0)$. It is easy for each player $P_i$ to compute their new share $\hat{f}(i)$:

$$\hat{f}(i) = f(j) - j\left(\frac{f(i) - f(j)}{i - j}\right). \tag{4.11}$$

Therefore, each $P_i$ privately computes $\hat{f}(i)$ to update his share. As a result, the threshold is decreased while the secret remains the same. ∎

**Example 4.5** *Let $t = 3$ be the threshold and let $f(x) = 9 + 2x + 5x^2 \in \mathbb{Z}_{13}[x]$ be the secret sharing polynomial. The dealer creates the shares of players $P_1, P_2,$ and $P_3$ accordingly (i.e., $f(1) = 3, f(2) = 7, f(3) = 8$) and he leaves the scheme.*

*Suppose the players first collaborate to compute and reveal $f(4)$. They then combine this share with their own private shares to decrease the threshold. Similar to Example 2.3 in Chapter 2, the players first compute and then they reveal $\sigma_1 = 7, \sigma_2 = 4,$ and $\sigma_3 = 8$ publicly (in Example 2.3, the players revealed $\sigma_j$ only to $P_4$). They then add up these values to compute $f(4) = 6$. Finally, shares of the players are updated as follows:*

$$\hat{f}(1) = 6 - 4\left(\frac{3 - 6}{1 - 4}\right) = 2 \quad \hat{f}(2) = 6 - 4\left(\frac{7 - 6}{2 - 4}\right) = 8 \quad \hat{f}(3) = 6 - 4\left(\frac{8 - 6}{3 - 4}\right) = 1.$$

*The new secret sharing polynomial is $\hat{f}(x) = 9 + 6x$, having the same secret 9.*

## 4.4 Schemes in the Active Adversary Model

First, we show how re-sharing methods fail in the active adversary setting. We then provide two different solutions, for threshold increase and decrease respectively. Our solutions are based on the VSS scheme in Section 1.3.3, so we assume the number of malicious players is at most $t-1 \leq \left\lfloor \frac{n-1}{4} \right\rfloor$. Therefore, in addition to the secure private channels between each pair of players, a synchronous broadcast channel is also assumed to exist. All computations are again performed in a finite field $\mathbb{Z}_q$, where $q$ is a prime number.

### 4.4.1 Failure of the Re-sharing Method

We first consider an obvious variation of the previous re-sharing scheme (Figure 4.1) that attempts to increase the threshold in the active adversary setting; this protocol is shown in Figure 4.4 (the authors in [67] provide a similar construction for proactive secret sharing in the situation where the threshold remains unchanged). We show that this technique actually fails to change the threshold. Moreover, we clarify why this re-sharing technique is not secure against a *mobile* adversary in the active adversary setting.

In a VSS scheme, such as the one presented in Figure 1.2, when the dealer uses a symmetric polynomial to generate shares, each pair $P_i$ and $P_j$ can check the validity of their shares through private channels. The (symmetric) matrix containing these values is called a *pairwise check matrix*:

$$\mathcal{C}_{n\times n} = \begin{pmatrix} - & f_1(\omega^2) & \dots & f_1(\omega^n) \\ f_2(\omega^1) & - & \dots & f_2(\omega^n) \\ \vdots & \vdots & \ddots & \vdots \\ f_n(\omega^1) & f_n(\omega^2) & \dots & - \end{pmatrix} \tag{4.12}$$

where $f_i(\omega^j) = f_j(\omega^i)$ for all $i, j$.

Since our focus is to construct a dealer-free protocol for changing the threshold, we assume that the dealer who initiates the scheme is honest. Recall that the honest dealer first initiates the secret sharing scheme using a symmetric bivariate polynomial, i.e., he randomly generates $f(x, y) \in \mathbb{Z}_q[x, y]$ of degree at most $t-1$, where $f(0, 0) = \alpha$ is the secret. The dealer sends the following shares to $P_i$ for $1 \leq i \leq n$, and then leaves:

$$f_i(x) = f(x, \omega^i) \text{ where } \omega \text{ is a primitive element in } \mathbb{Z}_q. \tag{4.13}$$

We present the (flawed) re-sharing protocol in Figure 4.4 and an example of this approach is presented subsequently. The idea is that each player $P_i$ re-shares his share $f_i(x)$ using the VSS scheme, by choosing a symmetric bivariate polynomial of degree at most $t' - 1$ that yields $f_i(x)$ when $y = 0$. Then appropriate pairwise checks are performed and new shares are computed.

**Theorem 4.6** *The re-sharing protocol presented in Figure 4.4 fails to increase the threshold because the constant terms of the shares remain the same after re-sharing.*

**Proof.** In the basic VSS scheme, it suffices to use only the constant terms to reconstruct the secret. If a set $\Delta$ of at least $t$ honest players combine their values $f_j(0)$ ($j \in \Delta$), the secret $\alpha = f(0,0)$ is revealed. From the secret recovery steps in Figure 1.2, we have

$$\sum_{j \in \Delta} \left( \lambda_j^{\Delta} \times f_j(0) \right) = f(0,0).$$

We now prove that $\varphi_j(0) = f_j(0)$ for every $P_j$:

$$\begin{aligned}
\varphi_j(0) &= \sum_{i \in \Delta} \left( \lambda_i^{\Delta} \times g_i(0, \omega^j) \right) && \text{by (4.16)} \\
&= \sum_{i \in \Delta} \left( \lambda_i^{\Delta} \times g_i(\omega^j, 0) \right) && \text{by symmetry} \\
&= \sum_{i \in \Delta} \left( \lambda_i^{\Delta} \times f_i(\omega^j) \right) && \text{by (4.14)} \\
&= \sum_{i \in \Delta} \left( \lambda_i^{\Delta} \times f_j(\omega^i) \right) && \text{by symmetry} \\
&= f_j(0) && \text{by (1.3).}
\end{aligned}$$

Therefore, the threshold is not increased because $t$ players are able to reconstruct the secret after re-sharing (using the same constant terms as before). ∎

**Remark 4.7** Although handling a mobile adversary is out of the scope of this chapter, we should mention that the re-sharing technique (using bivariate polynomials) is not secure against a mobile adversary due to the same reason. That is, the mobile adversary can incrementally collect the constant terms of the players' shares in different time periods in order to recover the secret.

Threshold Modification

1. Each $P_i$ creates a symmetric polynomial $g_i(x, y)$ of degree $t' - 1$ such that
$$g_i(x, 0) = f_i(x). \tag{4.14}$$

2. Each player $P_i$ sends $g_i(x, \omega^j)$ to player $P_j$ for $1 \le i, j \le n$, i.e., re-sharing the original shares by auxiliary shares. The share-exchange matrix $\mathcal{E}_{n\times n}$, where each player generates a row and receives a column, is as follows:
$$\mathcal{E}_{n\times n} = \begin{pmatrix} g_1(x,\omega^1) & g_1(x,\omega^2) & \dots & g_1(x,\omega^n) \\ g_2(x,\omega^1) & g_2(x,\omega^2) & \dots & g_2(x,\omega^n) \\ \vdots & \vdots & \ddots & \vdots \\ g_n(x,\omega^1) & g_n(x,\omega^2) & \dots & g_n(x,\omega^n) \end{pmatrix}.$$

3. Players perform pairwise checks on $g_i(x, \omega^j)$, which can be shown by $n$ matrices. They also do a pairwise check on $g_i(0, \omega^j)$ to make sure that the constant terms of the shares are consistent with the shares distributed by the honest dealer.

4. A set $\Delta$ of size exactly $t$ is determined such that it contains the identifiers of $t$ good players (this set is defined after pairwise checks, see [26, 76] for details), and the following public Lagrange constants are computed:
$$\lambda_i^{\Delta} = \prod_{j\in\Delta, i\neq j} \frac{\omega^j}{\omega^j - \omega^i} \text{ where } 1 \le i, j \le n \text{ represent players' } ids. \tag{4.15}$$

5. Each player $P_j$ erases his old shares, and then combines the auxiliary shares he has received from other players to compute his new share as follows:
$$\varphi_j(x) = \sum_{i\in\Delta} \Big(\lambda_i^{\Delta} \times g_i(x, \omega^j)\Big). \tag{4.16}$$

Secret Recovery

- Now, if a set $\Delta'$ of at least $t'$ players cooperate, they can recover $\alpha$ by using the Lagrange interpolation method:
$$\alpha = \sum_{j\in\Delta'} \Big(\lambda_j^{\Delta'} \times \varphi_j(0)\Big). \tag{4.17}$$

Figure 4.4: Insecure Protocol For Threshold Modification in the Active Adversary Model

**Example 4.8** *Suppose that shares of $f(x,y) = \boldsymbol{9} + 4x + 4y + 7xy$ are distributed by the dealer among four players, where $t = 2$, we are working in the field $\mathbb{Z}_{13}$ and $\omega = 2$:*

$$\begin{aligned} f_1(x) &= f(x, 2^1) = \boldsymbol{4} + 5x & \qquad f_3(x) &= f(x, 2^3) = \boldsymbol{2} + 8x \\ f_2(x) &= f(x, 2^2) = \boldsymbol{12} + 6x & \qquad f_4(x) &= f(x, 2^4) = \boldsymbol{8} + 12x \end{aligned}$$

1. *Suppose the players re-share their shares using $g_i(x,y)$ for $1 \leq i \leq 4$, where $t' = 3$.*

$$\begin{aligned} g_1(x,y) &= (\boldsymbol{4} + 5x + 5y) + (3xy + 7xy^2 + 7x^2y + 5x^2y^2) \\ g_2(x,y) &= (\boldsymbol{12} + 6x + 6y) + (7xy + 8xy^2 + 8x^2y + 7x^2y^2) \\ g_3(x,y) &= (\boldsymbol{2} + 8x + 8y) + (8xy + 5xy^2 + 5x^2y + 3x^2y^2) \\ g_4(x,y) &= (\boldsymbol{8} + 12x + 12y) + (4xy + 9xy^2 + 9x^2y + 2x^2y^2) \end{aligned}$$

2. *Matrix $\mathcal{E}_{4\times4}$, where each $P_i$ generates a row and receives a column, is as follows:*

$$\begin{pmatrix} 1 + 8x^2 & 11 + 12x + 4x^2 & 5 + 9x + 12x^2 & 6 + 12x + x^2 \\ 11 + 5x^2 & 10 + 6x + x^2 & 8 + 2x + 5x^2 & 4 + 8x + 9x^2 \\ 5 + 5x + 9x^2 & 8 + 3x + 3x^2 & 1 + 2x + 11x^2 & 12x + 3x^2 \\ 6 + 4x & 4 + 3x + 3x^2 & 9x + 5x^2 & 5 + x + 6x^2 \end{pmatrix}.$$

3. *The players then perform pairwise checks on the shares that they have received. These matrices must be symmetric with respect to the main diagonal:*

$$\begin{pmatrix} 0 & 2 & 9 & 0 \\ 2 & 0 & 5 & 0 \\ 9 & 5 & 0 & 5 \\ 0 & 0 & 5 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 7 & 11 & 11 \\ 7 & 0 & 5 & 0 \\ 11 & 5 & 0 & 7 \\ 11 & 0 & 7 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 12 & 2 & 3 \\ 12 & 0 & 1 & 4 \\ 2 & 1 & 0 & 7 \\ 3 & 4 & 7 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 6 & 1 & 2 \\ 6 & 0 & 7 & 4 \\ 1 & 7 & 0 & 2 \\ 2 & 4 & 2 & 0 \end{pmatrix}$$

4. *The players define a set $\Delta$ which contains the identifiers of $t$ good players. Suppose $\Delta = \{P_1, P_2\}$; then we have*

$$\gamma_1 = \frac{(0-4)}{(2-4)} = 2 \qquad\qquad \gamma_2 = \frac{(0-2)}{(4-2)} = -1$$

5. *All the players update their shares and erase their old shares:*

$$\begin{aligned} \varphi_1(x) &= \boldsymbol{4} + 11x^2 & \qquad \varphi_3(x) &= \boldsymbol{2} + 3x + 6x^2 \\ \varphi_2(x) &= \boldsymbol{12} + 5x + 7x^2 & \qquad \varphi_4(x) &= \boldsymbol{8} + 3x + 6x^2 \end{aligned}$$

*Now, we show that $t = 2$ players, say $P_2, P_3$, can reconstruct the secret, even though the threshold is now supposed to be $t' = 3$, as follows:*

$$\alpha = \frac{(0-8)}{(4-8)}(\boldsymbol{12}) + \frac{(0-4)}{(8-4)}(\boldsymbol{2}) = \boldsymbol{9}.$$

### 4.4.2 Threshold Decrease by Public Evaluation

The original secret sharing polynomial $f(x, y) \in \mathbb{Z}_q[x, y]$ of degree at most $t - 1$ is a symmetric bivariate polynomial. Each player $P_i$ receives $f_i(x) = f(x, \omega^i) \in \mathbb{Z}_q[x]$ from the dealer. For the sake of simplicity, suppose the players want to decrease the threshold from $t$ to $t - 1$ in the active adversary model and in the absence of the dealer (for further threshold reduction, they can repeat the procedure). The players can first use the *recovery* protocol proposed in [98, Section 4.3] to publicly generate an extra share $f_j(x)$ at a desired point $\omega^j$ where $j$ is different from the existing players' *ids*. This is similar to what we did in Figure 4.3 in the passive adversary case. Next, the players need to combine the revealed share with their own private shares; this is more complicated now that we are in the active adversary model, due to the use of symmetric bivariate polynomials. The protocol to accomplish this is presented in Figure 4.5 and an example to illustrate these computations is presented subsequently.

**Theorem 4.9** *The threshold reduction protocol presented in Figure 4.5 is secure under the active adversary model tolerating $t - 1 \leq \left\lfloor \frac{n-1}{4} \right\rfloor$ colluders, and it correctly reduces the threshold.*

**Proof.** If at most $t-1$ malicious parties reveal incorrect values in the first step, the players can correctly compute $f_j(x)$ in the second step through an error correction technique (see Section 1.3.3 for details). It is also easy to show that all the bivariate polynomials remain symmetric during the third and fourth steps. As a result, the players can perform pairwise checks in order to detect any malicious behavior. We just need to show that the new shares are on a polynomial $\hat{f}(x, y)$ of degree at most $t - 2$, where $\hat{f}(0, 0) = f(0, 0)$. We know that

Threshold Decrease

1. The players randomly select a new *id* $j$ such that $j \notin \mathcal{P}$. Each player $P_i$ then computes and reveals $f_i(\omega^j)$ to all the other players.

2. Subsequently, the players compute the new share $f_j(x)$ by using points $(i, f_i(\omega^j))$ (see [98, Section 4.3]).

3. Each $P_i$ combines his private share $f_i(x)$ with the new share $f_j(x)$ as follows:

$$\hat{f}_i(x) = \hat{f}(x, \omega^i) = (\omega^j)^2 \left( \frac{f_i(x) - f_j(x) - f_j(\omega^i) + f_j(\omega^j)}{(x - \omega^j)(\omega^i - \omega^j)} \right) + 2f_j(0) - f_j(\omega^j). \tag{4.18}$$

4. Shares $\hat{f}_i(x) \in \mathbb{Z}_q[x]$ correspond to a symmetric bivariate polynomial $\hat{f}(x, y) \in \mathbb{Z}_q[x, y]$ of degree $t - 2$ that has the same secret as before. Therefore, $t - 1$ players are now sufficient to recover the secret.

Figure 4.5: Threshold Decrease by Public Evaluation in the Active Adversary Model

$$(y - \omega^j) \mid \big(f(x, y) - f_j(x)\big)$$

and

$$(y - \omega^j) \mid \big(f_j(y) - f_j(\omega^j)\big).$$

Now, it is clear that, if $a \mid b$ and $a \mid c$, then $a \mid (b - c)$. As a result, we obtain:

$$(y - \omega^j) \mid \big(f(x, y) - f_j(x) - f_j(y) + f_j(\omega^j)\big).$$

Since the right hand side of the above equation remains symmetric, $(x - \omega^j)$ also divides it. Now, if $a \mid c$, $b \mid c$, and $gcd(a, b) = 1$, then $ab \mid c$. Moreover, $(x - \omega^j)$ and $(y - \omega^j)$ do not have a common divisor. Therefore, we obtain

$$(x - \omega^j)(y - \omega^j) \mid \big(f(x, y) - f_j(x) - f_j(y) + f_j(\omega^j)\big).$$

Let define a symmetric polynomial $f^*(x, y)$ of degree $t - 2$ as follows:

$$f^*(x,y) \overset{\text{def}}{=} \frac{f(x,y) - f_j(x) - f_j(y) + f_j(\omega^j)}{(x-\omega^j)(y-\omega^j)}. \tag{4.19}$$

Accordingly, using (4.19), the secret associated with $f^*$ will be:

$$f^*(0,0) = \frac{f(0,0) - f_j(0) - f_j(0) + f_j(\omega^j)}{(\omega^j)^2}. \tag{4.20}$$

Using (4.20), the relation between the old secret and the new one is obtained:

$$f(0,0) = (\omega^j)^2 f^*(0,0) + 2f_j(0) - f_j(\omega^j). \tag{4.21}$$

Suppose we define another symmetric polynomial $\hat{f}(x,y)$, of degree at most $t-2$, as follows:

$$\begin{aligned}\hat{f}(x,y) &= (\omega^j)^2 f^*(x,y) + 2f_j(0) - f_j(\omega^j) && (4.22)\\ &= (\omega^j)^2\left(\frac{f(x,y) - f_j(x) - f_j(y) + f_j(\omega^j)}{(x-\omega^j)(y-\omega^j)}\right) + 2f_j(0) - f_j(\omega^j). && (4.23)\end{aligned}$$

From (4.23), we see that each player $P_i$ can privately compute $\hat{f}_i(x) = \hat{f}(x,\omega^i)$ to update his share:

$$\hat{f}(x,\omega^i) = (\omega^j)^2\left(\frac{f(x,\omega^i) - f_j(x) - f_j(\omega^i) + f_j(\omega^j)}{(x-\omega^j)(\omega^i-\omega^j)}\right) + f_j(x) + f_j(\omega^i) - f_j(\omega^j).$$

Furthermore, from (4.22) and (4.21), it is clear that $f(0,0) = \hat{f}(0,0)$, so the secret remains the same. ∎

**Example 4.10** *Let $t = 3$ be the threshold and let the secret sharing polynomial $f(x,y) \in \mathbb{Z}_{13}[x,y]$ be defined as $f(x,y) = 5 + 11x + 11y + 9x^2 + 9y^2 + xy^2 + x^2y + xy + 3x^2y^2$. The dealer creates shares $f_i(x,\omega^i)$ of $P_1, P_2, P_3$, where $\omega = 2$ is a primitive element in the field:*

$f_1(x) = 11 + 4x + 10x^2$ $\qquad$ $f_2(x) = 11 + 5x + 9x^2$ $\qquad$ $f_3(x) = 6 + 5x + x^2$

*Subsequently, players collaborate to reveal $f_4(x)$ publicly. They then combine this share with their own private shares in order to decrease the threshold. According to Figure 4.5, the procedure is as follows:*

1. *The players compute $f_i(\omega^j)$ where $j = 4$. They reveal $f_1(2^4) = 9$, $f_2(2^4) = 3$, and $f_3(2^4) = 4$ accordingly.*

2. *They then compute the new share $f_4(x) = 2 + 10x$ by interpolating points $(2^1, 9)$, $(2^2, 3)$, and $(2^3, 4)$. The pairwise check matrix for all four shares is as follows:*

$$\begin{pmatrix} \mathbf{0} & \mathbf{5} & \mathbf{7} & 9 \\ \mathbf{5} & \mathbf{0} & \mathbf{3} & 3 \\ \mathbf{7} & \mathbf{3} & \mathbf{0} & 4 \\ 9 & 3 & 4 & 0 \end{pmatrix}.$$

3. *Each $P_i$ combines his private share $f_i(x)$ with the revealed share $f_4(x)$ as follows:*

$$\begin{aligned}
\hat{f}_1(x) &= (2^4)^2\left(\frac{(11+4x+10x^2)-(2+10x)-(2+10(2^1))+(2+10(2^4))}{(x-2^4)(2^1-2^4)}\right) \\
&\quad + 2(2) - (2+10(2^4)) = x+3 \\
\hat{f}_2(x) &= (2^4)^2\left(\frac{(11+5x+9x^2)-(2+10x)-(2+10(2^2))+(2+10(2^4))}{(x-2^4)(2^2-2^4)}\right) \\
&\quad + 2(2) - (2+10(2^4)) = 3x+1 \\
\hat{f}_3(x) &= (2^4)^2\left(\frac{(6+5x+x^2)-(2+10x)-(2+10(2^3))+(2+10(2^4))}{(x-2^4)(2^3-2^4)}\right) \\
&\quad + 2(2) - (2+10(2^4)) = 7x+10
\end{aligned}$$

*After the share recombination, the pairwise check matrix is as follows:*

$$\begin{pmatrix} 0 & 7 & 11 \\ 7 & 0 & 12 \\ 11 & 12 & 0 \end{pmatrix}.$$

4. *Now two players are sufficient to interpolate $\hat{f}_i(x)$ and recover a new symmetric bivariate polynomial $\hat{f}(x, y)$ of degree one with the same secret 5 as before. Suppose the first two players want to recover the secret. They can compute:*

$$\begin{aligned}\hat{f}(x,y) &= \frac{y-2^2}{2^1-2^2}(x+3)+\frac{y-2^1}{2^2-2^1}(3x+1)\\ &= xy+12x+12y+\mathbf{5}.\end{aligned}$$

### 4.4.3 Threshold Increase by Zero Addition

The idea of the next protocol we present is to increase the threshold by constructing shares of a polynomial that corresponds to a secret having the value zero and threshold $t' > t$, and adding these new shares to the players' current shares. We call this threshold increase by *zero addition.* The first part of this process involves a straightforward protocol, which we call *polynomial production*, that enables the players to collectively generate shares of an unknown secret $\delta$ without a dealer. This idea can be found in various early papers (e.g., [5], but security against active adversaries is not discussed there). In our version of the protocol, shown in Figure 4.6, we assume that at most $t-1$ players are actively malicious. If we use the VSS scheme from Section 1.3.3, then we require that $t-1 \leq \left\lfloor \frac{n-1}{4} \right\rfloor$.

Polynomial Production

1. Initially, $t+1$ players $P_i$ are selected at random in order to act as independent dealers; they each might be honest or malicious.

2. Each $P_i$ shares a secret, say $\delta_i$, among all the players using a VSS scheme where the degree of the secret sharing polynomial is $t-1$. If the sharing is accepted, then all good players have consistent shares of the secret $\delta_i$.

3. Each $P_i$ adds shares of the accepted $\delta_i$'s together. As a result, each player $P_i$ has a share on a symmetric polynomial $g(x,y)$ of degree $t-1$ with a constant term $\delta = \sum \delta_i$.

Figure 4.6: Generating Shares of a Random Number $\delta$ in the Active Adversary Model

In the first step of polynomial production, each of $t+1$ participants $P_i$ acts as an independent dealer and distributes shares of a secret $\delta_i$ using a VSS. Each of these $P_i$'s may be honest or dishonest. However, a dishonest $P_i$ can successfully corrupt at most $t-1$

shares; otherwise, he will be disqualified. If a player is disqualified in the second step, the other players simply discard the shares that he has distributed.

Suppose there are $\tau_1$ dealers who are disqualified in step 2 and suppose there are $\tau_2$ dealers who are dishonest but who distribute shares of their secret correctly (note that $\tau_1 + \tau_2 \leq t - 1$ by assumption). Now, the final secret $\delta$ is the sum of $t + 1 - \tau_1$ accepted $\delta_i$'s. Under the assumptions of the VSS that is being used, we know that each accepted $\delta_i$ has been shared correctly and can ultimately be reconstructed.

It is possible that $\tau_2$ of the accepted $\delta_i$'s might be revealed (correctly or incorrectly) by malicious players. However, these players cannot prevent the successful reconstruction of $\delta$. Furthermore, there are at least $n + 1 - \tau_1 - \tau_2 \geq 2$ accepted $\delta_i$'s that have not been revealed. Therefore, $\delta$ remains secret to everyone even if $\tau_2$ malicious dealers reveal their $\delta_i$'s before reconstruction.

Suppose each player initially has a share of the secret $\alpha$ on a polynomial of degree $t-1$. We will show how to use the polynomial production protocol in order to increase the threshold *without changing the secret* $\alpha$. Our zero addition protocol for threshold increase is presented in Figure 4.7.

The trick used in this protocol is to adjust the (symmetric) polynomial $g(x, y)$ that is used to share the unknown secret value $\delta$ so that the participants instead have shares of 0. This can be done using a simple technique described in [26, p. 357]. If we define $\hat{g}(x, y) = (x + y)g(x, y)$, then $\hat{g}(0, 0) = 0$. Furthermore, each player can adjust his share in an appropriate way: $\hat{g}(x, \omega^i) = (x + \omega^i)g(x, \omega^i)$.

Threshold Increase

Suppose $f(x, \omega^i)$ is the share of $\alpha$ belonging to $P_i$ (see Figure 1.2).

1. Players use the VSS of [98] to generate shares of an unknown secret $\delta$ on a symmetric polynomial $g(x, y)$ of degree $t' - 2$ using polynomial production.
2. Each $P_i$ multiplies his share $g(x, \omega^i)$ by $(x + \omega^i)$. Now, each $P_i$ has a share of 0 on the symmetric polynomial $\hat{g}(x, y) = (x + y)g(x, y)$ of degree $t' - 1$.
3. Each player adds his share $f(x, \omega^i)$ of $\alpha$ to his share $(x + \omega^i)g(x, \omega^i)$ of 0. As a result, each player has a share of $\alpha$ where the new threshold is $t' > t$.

Figure 4.7: Increasing the Threshold by Zero Addition in the Active Adversary Model

After executing Protocol 4.7, the secret remains the same while the threshold is increased to $t'$. Since the polynomials remain symmetric during every single step of the proposed protocol, the players can detect any malicious behavior by pairwise checks through secure channels, as was done in the VSS presented in Figure 1.2.

**Remark 4.11** Threshold increase by zero addition can also be achieved in the *passive* adversary model by appropriately modifying the approach of Figure 4.7.

## 4.5 Changing Both the Threshold and the Secret

In this section, we discuss how to increase threshold $t$ to a new threshold $t' > t$ and simultaneously update the secret $\alpha$ to a new value $\alpha'$ that depends on $\alpha$ in some specified way, i.e., $\alpha' = \alpha\beta + \delta$ where $\beta, \delta \in \mathbb{Z}_q$ are unknown constants and $q$ is prime. The updating should take place in the absence of the dealer who initially created the scheme. The methods we describe are based on verifiable secret sharing schemes and thus they are secure in the active adversary setting. To motivate our ideas, let's first adapt some well-known "folklore" methods (e.g., [9]) to combine shares of two existing secrets to form shares of a new secret. However, note that we are using these techniques in conjunction with a threshold increase, which is not usually considered in most previous works. Here, we suppose that an unknown secret $\alpha$ has been shared among $n$ players using a Shamir scheme with threshold $t$. We have the following operations, where we assume $t' > t$:

- Let $\delta$ be an unknown secret that has been shared among the same $n$ players, again using a Shamir scheme with threshold $t'$. If every player computes the sum of their two shares, then the result is a sharing of $\alpha' = \alpha + \delta$. Here, the threshold value is increased to $t'$.
- Let $\beta$ be another unknown secret that has been shared among the same $n$ players, again using a Shamir scheme with threshold $t' - t + 1$. If every player computes the product of their two shares, then the result is a sharing of $\alpha' = \alpha\beta$. Here, the threshold value is also increased to $t'$. However, it should be noted that the resulting secret sharing polynomial of degree $t' - 1$ is not a "random" polynomial ([9]) because it can be factored into two smaller polynomials of degrees $t - 1$ and $t' - t$, a property that will not hold for "random" polynomials. Therefore, we should not use this product construction in isolation; we should always follow it by an addition operation that increases or maintains the threshold, to guarantee that the final secret sharing polynomial is random.

The above methods of forming shares of the sum or products of two secrets assume that the players already have shares of two secrets. Here, we consider a slightly different setting, where the players have shares of one secret, $\alpha$, and they want to obtain shares of a new secret, $\alpha' = \alpha\beta + \delta$. The values of $\beta$ and $\delta$ are unknown random numbers that are determined interactively by the set of players. Earlier, we showed how the players can collectively generate shares of an unknown secret $\delta$ without a dealer, shown in Figure 4.6. Once the players have shares of $\beta$ and $\delta$, they can first compute shares of $\alpha\beta$ using the product protocol described above. They can then compute shares of $\alpha\beta + \delta$ using the sum protocol described above. Here is a more detailed description of a process that will change the secret $\alpha$ to $\alpha' = \alpha\beta + \delta$, while simultaneously increasing the threshold from $t$ to $t'$.

1. Use polynomial production protocol to create shares of an unknown secret $\beta$ with threshold $t' - t + 1$.
2. Each player multiplies his share of $\alpha$ with his share of $\beta$. As a result, the new threshold is $t'$ (however, at this point, the secret sharing polynomial is not random).
3. Use polynomial production protocol to create shares of an unknown secret $\delta$ with threshold $t'$.
4. Each player adds his share of $\alpha\beta$ and his share of $\delta$. Now each player has a share of $\alpha'$ with threshold $t'$ (and the secret sharing polynomial is random).

### 4.5.1 Application: Sequential Secret Sharing

In this section, we present an application of dynamic threshold schemes (where the secret is changed based on an unknown linear combination of previous secrets) by describing a new scheme that we term *sequential secret sharing*. In this construction, players progressively construct a sequence of secret sharing schemes with different (but related) secrets and thresholds in the absence of the dealer, i.e., they will modify the threshold while generating multiple secrets. For the sake of simplicity, we just use the addition operation in order to change the secret. Let's first start with an example to make this protocol clear.

**Example 4.12** *Suppose the goal is to create a three-level sequential secret sharing scheme among a set of thirteen players. Consider the following subsets of players:*

$$\begin{aligned} \mathcal{P} &= \{P_1, \ldots, P_{13}\}, & \mathcal{P}_1 &= \{P_1, P_2, P_3\}, \\ \mathcal{P}' &= \{P_4, \ldots, P_{13}\}, & \mathcal{P}_2 &= \{P_4, P_5, P_6, P_7\}, \\ & & \mathcal{P}_3 &= \{P_8, P_9, P_{10}, P_{11}, P_{12}, P_{13}\}. \end{aligned}$$

*Secret Sharing*

1. *The dealer first shares a* master secret $\alpha_1$ *with the players in* $\mathcal{P}$ *using a* $(2, 13)$-*threshold scheme. We denote this sharing by the following notation:*

$$\alpha_1 : \mathcal{P} = \{P_1, \ldots, P_{13}\}^{t_0=2}.$$

2. (a) *The players* $P_i \in \mathcal{P}$ *use polynomial production to create shares of an unknown secret* $\beta_1$ *having a threshold* $t_1 = 3$.

   (b) *They add their shares locally to obtain the shares of* $\alpha_2 = \alpha_1 + \beta_1$ *which has a threshold of* $t_1 = 3$. *All the players erase the shares of* $\alpha_1$.

   (c) $\{P_1, \ldots, P_3\}$ *only keep the shares of* $\beta_1$, *and* $\{P_4, \ldots, P_{13}\}$ *keep the shares of* $\alpha_2$. *Using the notation defined above, the result is denoted by:*

$$\beta_1 : \mathcal{P}_1 = \{P_1, P_2, P_3\}^{t_1=3} \quad \text{and} \quad \alpha_2 : \mathcal{P}' = \{P_4, \ldots, P_{13}\}^{t_1=3}.$$

3. (a) *The players* $P_i \in \mathcal{P}'$ *use polynomial production to create shares of an unknown secret* $\beta_2$ *having a threshold* $t_2 = 4$.

   (b) *They add their shares locally to obtain the shares of* $\alpha_3 = \alpha_2 + \beta_2$ *which has a threshold of* $t_2 = 4$. *The players* $P_i \in \mathcal{P}'$ *erase the shares of* $\alpha_2$.

   (c) $\{P_4, \ldots, P_7\}$ *only keep the shares of* $\beta_2$, *and* $\{P_8, \ldots, P_{13}\}$ *increase the threshold from* $t_2 = 4$ *to* $t_3 = 6$ *and keep the shares of* $\alpha_3$. *We denote this by:*

$$\beta_2 : \mathcal{P}_2 = \{P_4, \ldots, P_7\}^{t_2=4} \quad \text{and} \quad \alpha_3 : \mathcal{P}_3 = \{P_8, \ldots, P_{13}\}^{t_3=6}.$$

*Secret Recovery*

1. *In the first step, the players* $\mathcal{P}_3 = \{P_8, \ldots, P_{13}\}$ *recover the secret* $\alpha_3$.

2. *Subsequently,* $\mathcal{P}_2 = \{P_4, \ldots, P_7\}$ *recover the secret* $\beta_2$. *As a result,* $\alpha_2$ *is revealed since* $\alpha_3 = \alpha_2 + \beta_2$.

3. *Finally,* $\mathcal{P}_1 = \{P_1, \ldots, P_3\}$ *recover the secret* $\beta_1$. *As a result, the master secret* $\alpha_1$ *is revealed since* $\alpha_2 = \alpha_1 + \beta_1$.

*As a realization of this example (hierarchical authority), we could imagine the president and vice president as the set $\mathcal{P}_1$, ministers as the set $\mathcal{P}_2$, and senators as the set $\mathcal{P}_3$ accordingly. Any decision on troops by the president and vice president is subject to the confirmations of ministers and senators. On the other hand, even having those confirmations, the final decision is made by the president and vice president, i.e., recovering the master secret $\alpha_1$.*

We now provide the definition of sequential secret sharing and then we present our protocol in Figure 4.8. Observe that Example 4.12 has $l = 3$ levels, and thresholds $t_0 = 2$, $t_1 = 3$, $t_2 = 4$, and $t_3 = 6$.

**Definition 4.13** Sequential secret sharing *is a secret sharing scheme where secrets $\alpha_1, \dots,$ $\alpha_l$ are shared among the players with monotonically increasing thresholds $t_0 < t_1 < \cdots < t_l$. Let $\mathcal{P}$ be a set of $n$ players and assume $\mathcal{P}$ is composed of $l$ disjoint levels*

$$\mathcal{P} = \bigcup_{i=1}^{l} \mathcal{P}_i,$$

*where $\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$ for all $1 \leq i < j \leq l$ and $|\mathcal{P}_i| \geq t_i$ for all $i$. The secret $\alpha_k$ (in level $k$) can be then recovered only if players in $\mathcal{R}_k = \bigcup_{i=k}^{l} \mathcal{P}_i$ cooperate and first recover their secrets sequentially, i.e., from the highest level $l$ down to level $k$.*

### Comparison with Existing Hierarchical Secret Sharing Schemes

Simmons [95] first proposed *disjunctive multilevel secret sharing*, and then Tassa [103] extended that construction to *conjunctive multilevel secret sharing*. They both use a single secret to construct their hierarchical threshold secret sharing schemes, whereas we generate several secrets in our access structure.

To explain how our *sequential secret sharing* (as shown in Figure 4.8) differs from these schemes, consider Example 4.12, where $t_1 = 3$, $t_2 = 4$, and $t_3 = 6$. In the case of the *disjunctive multilevel secret sharing*, players from $\mathcal{P}_1$ can recover the secret without the contributions of other players, i.e., cooperations of all levels are not required. In the case of the *conjunctive multilevel secret sharing*, all the players from $\mathcal{P}_1$, one player from $\mathcal{P}_2$ and two players from $\mathcal{P}_3$ can recover the secret. In both cases, there exists only a single secret, whereas, in our sequential secret sharing, we have several secrets.

Secret Sharing

1. A dealer uses a Shamir scheme to distribute shares of an initial secret $\alpha_1$ with threshold $t_0$ among players $\mathcal{P} = \{P_1, \ldots, P_n\}$, and then leaves the scheme.

2. Subsequently, players perform the following steps for $1 \le i \le l-1$:

   (a) The players in $\mathcal{P}$ use polynomial production to generate shares of a random secret $\beta_i$ with threshold $t_i$, where $t_{i-1} < t_i$.

   (b) They compute shares of $\alpha_{i+1} = \alpha_i + \beta_i \bmod q$; the threshold of $\alpha_{i+1}$ is $t_i$. Then they erase their shares of $\alpha_i$.

   (c) A subset of players, say $\mathcal{P}_i \subset \mathcal{P}$ where $|\mathcal{P}_i| \ge t_i$, only keep shares of $\beta_i$ and the rest of the players, i.e., $\mathcal{P} - \mathcal{P}_i$, only keep shares of $\alpha_{i+1}$.

   (d) If $i = l-1$ (i.e., the last step of the protocol is being executed), they increase the threshold from $t_{l-1}$ to $t_l$. Otherwise (if $i < l-1$), they set $\mathcal{P} \leftarrow \mathcal{P} \backslash \mathcal{P}_i$.

Secret Recovery

1. Appropriate subsets of players first cooperate to recover $\alpha_l$ and $\beta_{l-1}, \ldots, \beta_1$.

2. They then solve the following system of linear congruences: $\alpha_{i+1} \equiv \alpha_i + \beta_i \bmod q$ for $i = l-1$ down to $i = 1$. (It is clear that each congruence has a unique solution for $\alpha_i$ given $\alpha_{i+1}$ and $\beta_i$.) Therefore, $\alpha_l, \ldots, \alpha_1$ are recovered.

Figure 4.8: Sequential Secret Sharing Scheme

## 4.6 Conclusion

In this chapter, various *dynamic threshold schemes* were discussed. We illustrated how to increase or decrease the threshold in the passive and active adversary models when the dealer is not involved in the scheme after the scheme's initialization. In addition, we showed how to change the secret and increase the threshold at the same time. Finally, *sequential secret sharing* was proposed as a new application of the dynamic threshold schemes. Table 4.1 summarizes secure threshold modification techniques in the passive and active adversary models.

| Threshold | Passive Adversary | Active Adversary |
|---|---|---|
| Decrease | Figure 4.1: Lagrange Method<br>Figure 4.2: Vandermonde Matrix<br>Figure 4.3: Public Evaluation | Figure 4.5: Public Evaluation |
| Increase | Figure 4.1: Lagrange Method<br>Figure 4.2: Vandermonde Matrix<br>Remark 4.11: Zero Addition | Figure 4.7: Zero Addition |

Table 4.1: Summary of Secure Threshold Modification Techniques

# Chapter 5

# Multicomponent Commitment

In this chapter, we develop a new multicomponent commitment scheme in order to construct *sealed-bid auction protocols* (*SAP*) similar to the proposed solutions in [91, 90, 99]. Our protocols are auctioneer-free and unconditionally secure whereas previous protocols rely on computational assumptions and use auctioneers. We first propose a *multicomponent commitment scheme* (*MCS*), that is, a commitment scheme with multiple committers and verifiers. Later, three secure first-price auction protocols are proposed, each of which has its own properties.

## 5.1 Introduction

Commitment schemes were introduced by Blum [12] in order to solve the coin flipping problem. In a commitment scheme, the first party initially commits to a value while keeping it hidden; that is the *commitment phase*. Subsequently, he reveals the committed value to the second party such that it can be verified; that is the *reveal phase*. Our intention is to construct an unconditionally secure commitment scheme suitable for secure auction settings where we have multiple committers, i.e., the bidders who commit to their sealed-bids, and multiple verifiers.

Rivest [88] proposed an unconditionally secure commitment scheme in which the sender and receiver are both computationally unbounded. He assumes the existence of a trusted initializer, Ted, and a secure private channel between each pair of parties. We quickly review this protocol in Figure 5.1 since it is the only commitment scheme in the literature with unconditional security at both "commitment" and "reveal" phases. All operations are in $\mathbb{Z}_q$ where $q$ is a prime number.

Three-Step Commitment Scheme

1. **Initialize:** Ted randomly selects $a$ and $b$ which define a line $y = ax + b$, where $a \in \mathbb{Z}_q^*$ and $b \in \mathbb{Z}_q$, and securely sends values $a$ and $b$ to Alice. He then selects a point $(x_1, y_1)$ on this line and sends it to Bob securely.
2. **Commit:** Alice computes $y_0 = ax_0 + b$ as a commitment and sends it to Bob, where $x_0$ is her secret.
3. **Reveal:** Alice discloses the pair $(a, b)$ as well as $x_0$ to Bob. Finally, Bob checks that points $(x_0, y_0)$ and $(x_1, y_1)$ are both on the line $y = ax + b$. If so, Bob accepts $x_0$, otherwise, he rejects it.

Figure 5.1: Unconditionally Secure Commitment Scheme with a Trusted Initializer

As stated in [13], there exists a minor problem with this scheme. In a scenario where $y_0 = y_1$ (e.g., the committed value $y_0$ is equal to the second value that Bob receives from Ted), Bob learns $x_0$ before the reveal phase. That is, if $y_0 = y_1$ then $x_0 = x_1$, because $y = ax + b$ is a one-to-one function. This problem is fixed in [13] by replacing $y_0 = ax_0 + b$ with $y_0 = x_0 + a$ in the commitment phase.

**Definition 5.1** *A commitment scheme satisfies the following two properties, where Alice is the "committer" and Bob is the "verifier":*

1. *Binding: Alice must not be able to alter her commitment after she has made it.*
2. *Hiding: Bob must not be able to find out the commitment without Alice revealing it.*

We briefly review computation costs of polynomial evaluation and interpolation for complexity analysis of our protocols. Using a naive approach to evaluate $f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$ at a single point $\alpha$, we need $3n - 4$ operations in the finite field. First we require $n - 2$ multiplications to compute $\alpha^2 = \alpha \times \alpha, \alpha^3 = \alpha \times \alpha^2, \ldots, \alpha^{n-1} = \alpha \times \alpha^{n-2}$. Then, computing terms $a_ix^i$ requires a further $n - 1$ multiplications. Finally, adding all terms together takes $n - 1$ additions. (This approach can be improved slightly by Horner's method but it will not change the complexity of the operation.) Therefore, the total cost of the evaluation for a polynomial of degree at most $n - 1$ at a single point $\alpha$ is $O(n)$, consequently, the evaluation at $n$ points takes $O(n^2)$. To interpolate $n$ points and construct a polynomial of degree at most $n-1$, we need $O(n^2)$ operations using Lagrange/Newton interpolation [34].

These techniques can be improved by using the fast multipoint evaluations ($n$ points) and the fast interpolation of a polynomial of degree at most $n-1$. These methods take $O(C(n)\log n)$, where $C(n)$ is the cost of multiplying two polynomials of degree at most $n-1$. Therefore, the multipoint evaluation and the fast interpolation take $O(n\log^2 n)$ arithmetic operations using fast Fourier transform, which requires the existence of a primitive root of unity in the field.

$$C(n) : \begin{cases} O(n^2) \text{ classical method} \\ O(n^{1.59}) \text{ Karatsuba' s method} \\ O(n\log n) \text{ Fast Fourier Transform} \end{cases}$$

### 5.1.1 Motivation

The growth of e-commerce technologies has created a remarkable opportunity for sealed-bid auction protocols in which bidders' valuations are kept private while defining the auction outcomes, that is, the winner and selling price. The main motivation for privacy is the fact that bidders' valuations can be used in future auctions and negotiations by different parties, say auctioneers, to maximize their revenues, or competitors to win the auction. As an example, suppose a bidder proposes his bid on a specific item. If this valuation is released and the bidder loses the auction, then the other parties can use this information in future auctions (or negotiations) of similar items. This problem can be resolved by creating privacy-preserving protocols for the determination of the auction outcomes.

Most of the current sealed-bid auction protocols are not unconditionally secure, i.e., their security relies on computational assumptions such as hardness of factoring or discrete logarithm. Our motivation therefore is to propose an unconditionally secure multicomponent commitment scheme for the construction of new secure first-price auction protocols. Our intention is to enforce the verifiability in the sense that all the parties can verify the correctness of the auction outcomes.

In our protocols, bidders first commit to their bids before the auction starts. They then apply a decreasing price mechanism to define the winner and selling price. That is, each protocol starts with the highest price and decreases the price step by step until the auction outcomes are determined. This is similar to the approach in [91, 90, 99].

The authors in [91] use *undeniable signature schemes*, in [90] they apply *public-key encryption schemes*, and in [99] they use *collision intractable random hash functions*. To show how our constructions differ from these solutions, we can highlight the following improvements. First, previous solutions are only computationally secure whereas our protocols are unconditionally secure. Second, they use an auctioneer to define auction outcomes whereas our protocols only use a trusted initializer.

The main difference between all the stated constructions and the *Dutch-style auction* (where the auction begins with a high price which is lowered until a buyer accepts it) is the early commitments where bidders decide on their bids ahead of time and independent of whatever information they may gain during the auction. Moreover, bidders cannot change their minds while the auction is running. Finally, we can better deal with a *rushing attack*. In an electronic Dutch-style auction, it might be difficult to determine which of two bids received in close succession was actually made first. Therefore, a malicious bidder might be able to wait and bid immediately after the bid of another party in order to win the auction with a best possible price. Our solution prevents such a rushing attack.

### 5.1.2 Contributions

As our main contribution, we initially construct a *multicomponent commitment scheme* where multiple committers and verifiers act on many secrets. After that, several unconditionally secure first-price auction protocols are constructed based on this new commitment scheme. Each of these protocols consists of a trusted initializer and $n$ bidders. The security holds under the honest majority assumption.

Our first construction is *a verifiable protocol without the non-repudiation property.* This protocol has a low computation cost. Our second construction is *a verifiable protocol with the non-repudiation property.* The computation cost of this protocol has an extra multiplicative factor based on the price range. Our last construction is *an efficient verifiable protocol with the non-repudiation property and partial privacy.* This protocol preserves the privacy of losing bids by a security relaxation with a lower computation cost.

### 5.1.3 Organization

This chapter is organized as follows. Section 5.2 provides the required preliminaries for further technical discussions. Section 5.3 reviews the literature of the sealed-bid auctions. Section 5.4 present our new commitment scheme in an unconditionally secure setting.

Section 5.5 provides three sealed-bid first-price auction protocols based on our commitment scheme. Finally, Section 5.6 provides concluding remarks.

## 5.2 Preliminary: Sealed-Bid Auctions

In an auction mechanism, the winner is a bidder who submitted the highest bid. To define the selling price, there exists two major approaches: *first-price auction* and *second-price auction*. In the former, the winner pays the amount that he has proposed, i.e., the highest bid. In the latter, the winner pays the amount of the second-highest bid.

There are other types of auctions such as $(M+1)$*st-price auction* and *combinatorial auctions*. In the former, the highest $M$ bidders win the auction and pay a uniform price defined by the $(M+1)$-price. The second-price auction is a special case of this type of auction where $M = 1$. In the latter, multiple items with interdependent values are sold at the same time while bidders are able to bid on any combination of items.

In the first-price auction, a bidder potentially is able to define the winner as well as the selling price at the same time. On the other hand, in the second-price auction, a bidder potentially is able to define either the winner or the selling price for the winner. As a consequence, he proposes the actual highest value, say $\kappa$, he can afford to pay, which is also a profitable price for him [104]. Suppose the proposed bid is less than $\kappa$. In this case, the bidder decreases his chance of winning. If the proposed bid is bigger than $\kappa$, the bidder might win with an unprofitable price.

This property of the second-price auction forces bidders to propose their true valuations; this is an interesting and useful characteristic. At the same time, there are some deficiencies with second-price auctions. For instance, bidders may be reluctant to reveal their actual valuations, or the auctioneer may use a fake second-highest bid for more revenue [89, 92].

Sealed-bid auction models have many fundamental characteristics [85]. Some of the most important properties are as follows:

- *Correctness*: determining the winner and selling price correctly.
- *Privacy*: preventing the release of private bids, that is, losing bids.
- *Verifiability*: verifying auction outcomes by bidders and auctioneers.
- *Non-Repudiation*: preventing bidders from denying their bids.

## 5.3 Previous Works: Sealed-Bid Auction Protocols

In the initial construction of the *first-price sealed-bid auction* protocols, the authors in [31] implemented a secure auction service by using verifiable secret sharing as well as verifiable signature sharing. At the end of the bidding time, auctioneers open bids to define outcomes, therefore, they all know the losing bids. Subsequently, various types of secure auction protocols were proposed in the literature.

The proposed first-price auction protocol in [50] (which is modified in [51]) demonstrated a multi-round scheme in which winners from an auction round take part in a subsequent tie-breaking round. The authors used the addition operation of multiparty computation in a passive adversary model. Later, the authors in [84] detected some shortcomings in this scheme, such as the lack of verifiability. They then improved the scheme in various ways.

We can mention other first-price secure auction protocols with computational security. In [83], the authors designed a sealed-bid auction protocol based on a homomorphic secret sharing scheme. Their construction relies on the hardness of computational problems and does not depend on any trust. They also showed that the proposed protocol is secure against different kinds of attacks. In [46], the authors applied secure function evaluation via ciphertexts and presented a Mix-and-Max auction protocol. In [18], the authors used homomorphic encryption such as the ElGamal cryptosystem to construct various cryptographic auction protocols.

The authors in [42] presented protocols for sealed-bid auctions using secure distributed computation. The bidders' valuations are never revealed to any party, even after the auction is completed. Their protocol supports first-price and second-price auctions. The general idea of their approach is to compare bids digit by digit using secret sharing techniques. This protocol is expensive from computational and communication points of view.

The authors in [49] presented a protocol for the $(M + 1)$*st-price auction*, where the highest $M$ bidders win the auction and pay a uniform price. They used a new method in which bidders' valuations are encoded by the degree of many distributed polynomials. The construction requires only two rounds of computations; one round for the bidders and one round for the auctioneers. This auction protocol utilizes the verification approach proposed in [82]. The proposed scheme in [16] is a fully private $(M + 1)$st-price auction protocol in which only the winners and the seller learn the selling price. This construction also applied verifiable secret sharing of [82]. It has two major shortcomings. First, the scheme is not able to handle ties among multiple winners. Second, it is not an efficient construction.

Finally, the authors in [19, 20] investigated the possibility of unconditional *full privacy* (i.e., considering a collusion of size $n-1$ among $n$ bidders without using any auctioneers) in sealed-bid auctions. They demonstrated that first-price sealed-bid auctions can be achieved with full privacy; however, the protocol's round complexity is exponential in the bid size. On the other hand, they proved the impossibility of full privacy for second-price sealed-bid auctions having more than two bidders. For secure second-price and $(M+1)$st-price auction protocols using other cryptographic techniques, see [46, 17, 1, 56, 77, 18]. For secure *combinatorial auction* protocols, where multiple items with interdependent values are sold simultaneously, see [100, 109, 110, 81].

## 5.4 Multicomponent Commitment Scheme

We initially provide the formal definition of a multicomponent commitment scheme, that is, a construction with several commitments. We also assume that all participants are computationally unbounded.

**Definition 5.2** *A multicomponent commitment scheme has multiple committers as well as verifiers, and is said to be unconditionally secure if the following conditions hold:*

1. ***Hiding**: each receiver cannot learn anything about the secrets before the reveal phase except with a small probability $\epsilon_1$.*
2. ***Binding**: each sender cannot cheat (with the colluders' help) in the reveal phase by sending a fake secret except with a small probability $\epsilon_2$.*
3. ***Validating**: assuming the sender is honest, the other honest players should be able to correctly validate each secret during the reveal phase in the presence of the colluders.*

In the following constructions, we have $n$ players $P_1, P_2, \ldots, P_n$, as well as a trusted initializer $T$ who leaves the scheme before starting the protocols. We consider the existence of a secure private channel between each pair of parties, and an authenticated public broadcast channel. We also assume the majority of players are honest.

For the sake of simplicity, first a scheme with one committer, say $P_i$, and several verifiers $P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n$ is presented in Figure 5.2. We then extend our approach to a protocol with multiple committers and several verifiers (i.e., $n$ independent instances of the previous scheme), as shown in Figure 5.3.

Multicomponent Commitment Scheme

1. **Initialize:** $T$ randomly selects $g(x) \in \mathbb{Z}_q[x]$ of degree $n-1$ (where $q$ is prime), and privately sends $g(x)$ to committer $P_i$. He then selects $n-1$ distinct points $(x_j, y_j)$ uniformly at random (without replacement) on this polynomial, and sends $(x_j, y_j)$ to $P_j$ (for $1 \leq j \leq n$ and $j \neq i$) through private channels:

$$y_1 = g(x_1), \quad y_2 = g(x_2), \quad \ldots, \quad y_n = g(x_n).$$

2. **Commit:** Player $P_i$ first selects the secret $x_i$ and computes $y_i = g(x_i)$ as a committed value. He then broadcasts $y_i$ to the other players.

3. **Reveal:** $P_i$ discloses the polynomial $g(x)$ and his secret $x_i$ to the other parties through the public broadcast channel. First of all, the other players verify that $y_i = g(x_i)$, where $y_i$ is the value that $P_i$ has already committed to. After that, each $P_j$ checks to see if his point is on $g(x)$, i.e., $y_j = g(x_j)$ for $1 \leq j \leq n$ and $j \neq i$. If $y_i = g(x_i)$ and the majority of players confirm the validity of $g(x)$ (or an equal number of confirmations and rejections are received), then $x_i$ is accepted as the secret of $P_i$, otherwise, it is rejected.

Figure 5.2: Unconditionally Secure MCS with a Trusted Initializer

**Theorem 5.3** *The multicomponent commitment scheme shown in Figure 5.3 is an unconditionally secure protocol under the honest majority assumption in an active adversary setting, i.e., it satisfies the hiding, binding, and validating properties of Definition 5.2 with* $\epsilon_1 = n/q$ *and* $\epsilon_2 = n^2/q$.

**Proof.** We will provide the proof when $n-1$ is even; the proof when $n-1$ is odd is similar. Malicious participants might be able to provide fake polynomials and consequently reveal incorrect secrets, or disrupt the voting result.

**Hiding**: when a player $P_i$ commits to a value, each player $P_j$ for $1 \leq j \leq n$ and $j \neq i$ only knows his pair $(x_{ij}, y_{ij})$ and the committed value $y_i$ in the first two phases. In the worst case scenario, even if $\frac{n-1}{2}$ players $P_j$ collude, they are not able to construct $g_i(x)$ of degree $n-1$ to reveal $x_i$. In the case where the committed value $y_i$ of $P_i$ is equal to some $y_{ij}$, $P_j$ might be able to infer some information about the secret $x_i$. This occurs with the following probability:

Generalized Multicomponent Commitment Scheme

1. **Initialize:** $T$ randomly selects $n$ polynomials $g_1(x),\ g_2(x),\ \ldots,\ g_n(x) \in \mathbb{Z}_q[x]$ of degree $n-1$, and privately sends $g_i(x)$ to $P_i$ for $1 \leq i \leq n$. He then selects $n-1$ distinct points $(x_{ij}, y_{ij})$ uniformly at random (without replacement) on each polynomial $g_i(x)$, and sends $(x_{ij}, y_{ij})$ to $P_j$ for $1 \leq j \leq n$ and $j \neq i$ through private channels. The following matrix shows the information that each player $P_j$ receives, i.e., all entries in the $j^{th}$ row:

$$\mathcal{E}_{n\times n} = \begin{pmatrix} g_1(x) & y_{21} = g_2(x_{21}) & \ldots & y_{n1} = g_n(x_{n1}) \\ y_{12} = g_1(x_{12}) & g_2(x) & \ldots & y_{n2} = g_n(x_{n2}) \\ \vdots & \vdots & \ddots & \vdots \\ y_{1n} = g_1(x_{1n}) & y_{2n} = g_2(x_{2n}) & \ldots & g_n(x) \end{pmatrix}.$$

2. **Commit:** each player $P_i$ computes $y_i = g_i(x_i)$ as a committed value and broadcasts $y_i$ to the other players, where $x_i$ is the secret of player $P_i$. In other words, $y_1, y_2, \ldots, y_n$ are the committed values and $x_1, x_2, \ldots, x_n$ are the secrets of the players accordingly.

3. **Reveal:** each player $P_i$ discloses $g_i(x)$ and his secret $x_i$ to the other parties through the public broadcast channel.

   (a) First, the other players verify that $y_i = g_i(x_i)$, where $y_i$ is the value that $P_i$ has already committed to.

   (b) In addition, they check to see if the $n-1$ points corresponding to $g_i(x)$ are in fact on this polynomial (that is, $y_{ij} = g_i(x_{ij})$ for $1 \leq j \leq n$ and $j \neq i$).

   If $y_i = g_i(x_i)$ and the majority of players confirm the validity of $g_i(x)$ (or an equal number of confirmations and rejections are received), then $x_i$ is accepted as a secret, otherwise, it is rejected.

Figure 5.3: Generalization of the Multicomponent Commitment Scheme

$$\mathbf{Pr}[y_i = y_{ij}] \leq \frac{n-1}{q} \quad \textit{for some } j \in [1,n] \textit{ and } j \neq i. \tag{5.1}$$

Consequently, with the probability $\epsilon_1 = \mathbf{Pr}[y_i = y_{ij} \wedge x_i = x_{ij}]$, player $P_j$ may know the secret $x_i$ before the reveal phase, where

$$\epsilon_1 \leq \frac{n-1}{q} \quad \text{by (5.1).}$$

**Binding**: if a player $P_i$ changes his mind and decides to cheat by revealing a fake secret $x'_i$, he needs to provide a fake polynomial $g'_i(x)$ of degree $n-1$ such that $y_i = g'_i(x'_i)$, since he has already committed to $y_i$, and $y_{ij} = g'_i(x_{ij})$ for $1 \leq j \leq n$ and $j \neq i$, meaning that $g_i(x)$ and $g'_i(x)$ must pass through $n-1$ common points, that is, $P_i$ needs to guess all points of the other players. The alternative solution for $P_i$ is to collude with malicious players and change the voting result such that a sufficient number of players accept the fake secret $x'_i$. It is clear that two distinct polynomials $g_i(x)$ and $g'_i(x)$ of degree $n-1$ agree at most on $n-1$ points, therefore, for a randomly selected point $(x_{ij}, y_{ij})$ we have:

$$\mathbf{Pr}[y_{ij} = g_i(x_{ij}) \wedge y_{ij} = g'_i(x_{ij})] \leq \frac{n-1}{q}. \tag{5.2}$$

Suppose we have the maximum number of colluders to support $P_i$ and assume $n-1$ is an even number. To satisfy the honest majority assumption, there are always two more honest voters, that is,

$$\left(\frac{n-1}{2} + 1\right) - \left(\frac{n-1}{2} - 1\right) = 2.$$

Since the committer $P_i$ is dishonest, he can only change the voting result if he guesses at least one point of honest players, which leads to an equal number of confirmations and rejections. As a consequence, the probability of cheating with respect to the binding property is as follows:

$$\epsilon_2 \leq \left(\frac{n-1}{2} + 1\right) \times \left(\frac{n-1}{q}\right) \leq \frac{n^2}{q} \quad \text{by (5.2).}$$

**Validating**: suppose the committer $P_i$ is honest and $n-1$ is an even number. To satisfy the honest majority assumption, there is an equal number of honest and dishonest voters

$P_j$ for $1 \leq j \leq n$ and $j \neq i$. Therefore, $g_i(x)$ and consequently $x_i$ are accepted since an equal number of confirmations and rejections is achieved. ∎

**Theorem 5.4** *The multicomponent commitment scheme presented in Figure 5.3 takes three rounds of communications and $O(n^2 \log^2 n)$ computation cost.*

**Proof.** It can be seen that every stage takes one round of communications which comes to three rounds in total. To achieve a better performance, suppose we use a primitive element $\omega$ in the field to evaluate polynomials, i.e., $y_{ij} = g_i(\omega^{x_{ij}})$. As a consequence, in the first two stages, each $g_i(x)$ of degree $n-1$ is evaluated at $n$ points with $O(n \log^2 n)$ computation cost. This procedure is repeated for $n$ polynomials, consequently, the total cost is $O(n^2 \log^2 n)$. In the third stage, everything is repeated with the same computation cost of the first two steps, therefore, the total computation cost is $O(2n^2 \log^2 n) = O(n^2 \log^2 n)$. ∎

## 5.5 Application: Sealed-Bid Auction Protocols

Now, three secure first-price auction protocols based on the multicomponent commitment scheme are presented. Our constructions are auctioneer-free in an unconditionally secure setting; i.e., bidders define auction outcomes themselves.

Our protocols consist of a trusted initializer $T$ and $n$ bidders $B_1, \ldots, B_n$ where bidders' valuations $\beta_i \in [\eta, \kappa]$. Let $\theta = \kappa - \eta$ denote the price range. An *initializer* leaves the scheme before running the protocols. This is preferable to a *trusted party* who remains in the scheme. In the literature, trusted parties are assumed in many secure auction protocols, for instance, semi-trusted third party [22, 7], trusted third party [107, 66], trusted centers [90], trusted authority [1, 101], trustee [105]. It is worth mentioning that by paying an extra computational and communication cost, a trusted party or initializer can be replaced by a secure multiparty computation protocol.

We assume the existence of secure private channels between the initializer and each bidder, as well as between each pair of bidders. There also exists an authenticated public broadcast channel on which information is transmitted instantly and accurately to all parties. Finally, we assume the majority of the bidders are honest, and at most $n/2$ of the bidders may collude to disrupt the auction outcomes or learn the losing bids. Let $\mathbb{Z}_q$ be a finite field, where $q$ is a prime number, and let $\omega$ be a primitive element in this field. All computations are performed in $\mathbb{Z}_q$. We need $n^2/q$ to be very small in order for the multicomponent commitment scheme to be secure, as stated in Theorem 5.3. Therefore, $q$ must be large enough to satisfy this requirement.

### 5.5.1 Verifiable Protocol with Repudiation Problem

The new protocol, which is shown in Figure 5.4, has many useful properties. First of all, it is a verifiable scheme in which bidders are able to verify the correctness of auction outcomes while preserving the privacy of losing bids. Second, it is a simple construction with a low computation cost. Finally, bidders are able to define auction outcomes without any auctioneers in an unconditionally secure setting. However, it has a shortcoming in the sense that a malicious player may refuse to acknowledge being the winner of the auction when his bid is equal to the current price $\gamma$, that is, it suffers from the *repudiation problem*.

**Theorem 5.5** *Excluding the repudiation problem, the auction protocol VR presented in Figure 5.4 determines auction outcomes correctly under the honest majority assumption with a small probability of error, as computed in Theorem 5.3, and it also protects the losing bids.*

**Proof.** Under the honest majority assumption and the proof in Theorem 5.3, the scheme protects the losing bids with a negligible probability of error and only reveals the highest bid. Moreover, bidders are able to verify the claim of the winner and consequently define the selling price with a negligible probability of cheating. It is worth mentioning that the protocol has definitely a winner since more than half of the players are honest. In other words, in the case of "repudiation" (where a malicious bidder refuses to claim his bid), the first honest bidder who claims as the winner is the ultimate winner of the auction. ∎

**Theorem 5.6** *The auction protocol VR takes at most $O(\theta)$ rounds of communications and $O(n^2 \log^2 n)$ computation cost.*

**Proof.** There exist two rounds of communication for the first two stages. In addition, the third phase takes at most $\theta$ rounds, which comes to $O(\theta)$ in total. To compute the computation cost, each $g_i(x)$ of degree $n-1$ is evaluated at $n$ points in the first two steps with $O(n \log^2 n)$ computation cost, i.e, $n-1$ evaluations in the first step and one evaluation in the second step. This procedure is repeated for $n$ bidders, as a consequence, the total cost is $O(n^2 \log^2 n)$. In the third stage, a constant number of polynomials equivalent to the number of winners are evaluated, therefore, the total computation cost for the entire protocol is $O(n^2 \log^2 n)$. Even if all players propose a common value and we have $n$ winners, the computation cost has the same complexity. ∎

VR: Sealed-Bid First-Price Auction

1. **Initialize:** The trusted initializer $T$ first provides the following private data through secure pair-wise channels, and then he leaves the scheme:

   (a) He randomly selects $n$ polynomials $g_1(x), g_2(x), \ldots, g_n(x) \in \mathbb{Z}_q[x]$ of degree $n-1$, and sends $g_i(x)$ to $B_i$ for $1 \leq i \leq n$.

   (b) He then selects $n-1$ distinct points $(\omega^{x_{ij}}, y_{ij})$ uniformly at random (without replacement) on each polynomial $g_i(x)$, and sends $(\omega^{x_{ij}}, y_{ij})$ to $B_j$ for $1 \leq j \leq n$ and $i \neq j$ through the secure channels.

2. **Start:** when the auction starts, each $B_i$ commits to his bid $\beta_i$ by computing $\alpha_i = g_i(\omega^{\beta_i})$ and broadcasting $\alpha_i$ to the other bidders. There is a specific time interval in which bidders are required to commit to their bids.

3. **Close:** after the closing time, bidders set the initial price $\gamma$ to be the highest possible price, i.e., $\gamma = \kappa$, and then define winners as follows:

   (a) The bidder $B_k$ who has committed to $\gamma$ claims that he is the winner. Consequently, he must prove $\beta_k = \gamma$. Ties among multiple winners can be handled by assigning a specified priority to the bidders or by a random selection after valid proofs by different winners have been given.

   (b) $B_k$ also reveals $g_k(x)$ so that the other bidders are able to verify that $\alpha_k = g_k(\omega^{\beta_k})$. They then check to see if all these $n-1$ points are on $g_k(x)$. If these conditions hold, then $B_k$ is accepted as the winner, otherwise, his claim is rejected.

   (c) If no one claims to be the winner or the bidder who claimed as a potential winner could not prove his plea, then bidders decrease the selling price by one, i.e., $\gamma = \kappa - 1$, and the procedure from stage (a) is repeated.

Figure 5.4: Verifiable Secure Auction Protocol with Repudiation Problem

### 5.5.2 Verifiable Protocol with Non-Repudiation

To handle the repudiation problem, we modify our earlier construction such that all the losers must prove that their bids are less than the winning price at the end of the protocol. This scheme is presented in Figures 5.5 and 5.6. We also provide an example of this scheme.

**Example 5.7** *Suppose each $\beta_i \in [0, 7]$ and all computations are performed in the field $\mathbb{Z}_{13}$ where $q = 13$. Assume $\beta_i = 7 - 5 = 2$ and the winning price is $\beta_k = 5$ or $\beta_k = 3$ in two different scenarios. Here $\mathcal{M}_i(x) \in [0, 7)$ if $x = 0$; otherwise, $\mathcal{M}_i(x) \in [7, 13)$. We then have the following possible vectors for the bids:*

$$\mathcal{B}_i = (1, 0, 1, 1, 0, 1, 1) \text{ and } \mathcal{B}'_i = (12, 6, 10, 7, 3, 11, 9).$$

- *Suppose the winning bid is $\beta_k = 5$. The loser $B_i$ reveals $7 - 5 + 1 = 3$ values larger than $q/2$ in order to prove he has at least three 1's in $\mathcal{B}_i$, which shows his bid is less than the winning price.*
- *Suppose the winning bid is $\beta_k = 3$. The loser $B_i$ reveals $7 - 3 + 1 = 5$ values larger than $q/2$ to prove that his bid is less than the winning price.*

**Remark 5.8** *By a simple modification, it is feasible to catch malicious bidders before determining the winner. In this modification, as we decrease the price one by one, each bidder $B_i$ must reveal one new $b'_{ij} \in [q/2, q)$ (i.e., $b_{ij} = 1$) at each round, otherwise, he is removed from the scheme.*

**Theorem 5.9** *The proposed auction protocol VNR presented in Figures 5.5 and 5.6 determines auction outcomes correctly under the honest majority assumption with a small probability of error, as computed in Theorem 5.3, and protects the losing bids. It also satisfies the non-repudiation property.*

**Proof.** We can follow the same proof in Theorem 5.3 for the verifiability and privacy. Moreover, it is required to show that losers do not reveal any unnecessary information about their bids in part $(c)$ of stage 3, beyond the fact that their bids are less than the winning bid. As shown in Figure 5.5, each bidder $B_i$ commits to $\theta$ values, where $\theta = \kappa - \eta$. According to Equation 5.5, a loser $B_l$ reveals $\kappa - \beta_k + 1$ commitments of 1's in part $(c)$ of stage 3. This has two meanings:

VNR-a: Sealed-Bid First-Price Auction

1. **Initialize:** The trusted initializer $T$ first provides some private data through secure pair-wise channels and then leaves the scheme:

   (a) He randomly selects $\theta$ polynomials $g_{i1}(x), g_{i2}(x), \ldots, g_{i\theta}(x) \in \mathbb{Z}_q[x]$ of degree $n-1$ for each $B_i$, and privately sends these polynomials to $B_i$ for $1 \leq i \leq n$, where $\theta$ is the price range.

   (b) He then selects $n-1$ distinct points $(\omega^{x^k_{ij}}, y^k_{ij})$ for $1 \leq k \leq n$ and $k \neq i$ uniformly at random on each $g_{ij}(x)$, where $1 \leq j \leq \theta$. He finally sends these points to $B_k$, i.e., each $B_k$ receives the entries in $k^{th}$ row of $\mathcal{E}_{n\times\theta n}$:

$$\mathcal{E}_{n\times\theta n} = \begin{pmatrix} g_{11}(x) & \ldots & g_{1\theta}(x) & \ldots & (\omega^{x^1_{n1}}, y^1_{n1}) & \ldots & (\omega^{x^1_{n\theta}}, y^1_{n\theta}) \\ (\omega^{x^2_{11}}, y^2_{11}) & \ldots & (\omega^{x^2_{1\theta}}, y^2_{1\theta}) & \ldots & (\omega^{x^2_{n1}}, y^2_{n1}) & \ldots & (\omega^{x^2_{n\theta}}, y^2_{n\theta}) \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ (\omega^{x^n_{11}}, y^n_{11}) & \ldots & (\omega^{x^n_{1\theta}}, y^n_{1\theta}) & \ldots & g_{n1}(x) & \ldots & g_{n\theta}(x) \end{pmatrix}.$$

2. **Start:** when the auction starts, there is a specific time interval in which bidders are allowed to commit to their bids.

   (a) Each bidder $B_i$ first defines his bid $\beta_i$ as follows:

$$\beta_i = \kappa - \sum_{j=1}^{\theta} b_{ij}. \tag{5.3}$$

   Here, $b_{ij} \in \{0,1\}$ and we write them as a vector $\mathcal{B}_i = (b_{i1}, b_{i2}, \ldots, b_{i\theta})$.

   (b) Each $B_i$ then applies a random mapping $\mathcal{M}_i(x) : \{0,1\} \to \mathbb{Z}_q$ to convert $\mathcal{B}_i$ to a new vector $\mathcal{B}'_i$ so that its elements $b'_{ij} \in \mathbb{Z}_q$. We require that $\mathcal{M}_i(x) \in [0, q/2)$ if $x = 0$; otherwise, $\mathcal{M}_i(x) \in [q/2, q)$.

   (c) Finally, each bidder $B_i$ for $1 \leq i \leq n$ commits to $b'_{ij}$ by $\alpha_{ij} = g_{ij}(\omega^{b'_{ij}})$ for $1 \leq j \leq \theta$ and broadcasts all $\alpha_{ij}$ to the other bidders.

Figure 5.5: a. Verifiable Secure Auction Protocol with Non-Repudiation

VNR-b: Sealed-Bid First-Price Auction

3. **Close:** after the closing time, bidders set the initial price $\gamma$ to be the highest possible price, i.e., $\gamma = \kappa$, and then define winners as follows:

   (a) $B_k$ who has committed to $\gamma$ claims he is the winner. Consequently, he must prove $\beta_k = \gamma$. Therefore, he reveals $g_{kj}(x)$ and $b'_{kj}$ for $1 \leq j \leq \theta$. By using the inverse mappings $[0, q/2) \to 0$ and $[q/2, q) \to 1$, $b_{kj}$ for $1 \leq j \leq \theta$ are recovered and $\beta_k$ is computed as follows:

   $$\beta_k = \kappa - \sum_{j=1}^{\theta} b_{kj}. \tag{5.4}$$

   (b) If $\beta_k = \gamma$, the other bidders then verify that $\alpha_{kj} = g_{kj}(\omega^{b'_{kj}})$ for $1 \leq j \leq \theta$. They also check to see if each set of $n-1$ points $(\omega^{x^i_{kj}}, y^i_{kj})$ for $1 \leq i \leq n$ and $i \neq k$ are on $g_{kj}(x)$'s. If these conditions hold, $B_k$ is accepted as the winner, otherwise, his claim is rejected.

   (c) Each loser $B_l$ must prove $\beta_l < \beta_k$. Therefore, each $B_l$ reveals any subset $\mathcal{J}$ of his commitments $b'_{lj}$ such that the following condition holds:

   $$\sum_{j \in \mathcal{J}} b_{lj} = \kappa - \beta_k + 1. \tag{5.5}$$

   where $b_{lj} = 1$ is the inverse mapping of $b'_{lj}$. Obviously, $B_l$ needs to provide valid proofs for $b'_{lj}$'s.

   (d) If no one claims to be the winner or the bidder who claimed to be the winner could not prove his plea, then bidders decrease the selling price by one, i.e., $\gamma = \kappa - 1$, and the procedure from stage (a) is repeated.

Figure 5.6: b. Verifiable Secure Auction Protocol with Non-Repudiation

1. Up to this time of the auction, the revealed portion of his bid is:

$$\begin{aligned}\beta_l &= \kappa - (\kappa - \beta_k + 1) \quad \text{by (5.3) and (5.5)}\\ &= \beta_k - 1.\end{aligned}$$

2. The number of commitments that have not been yet revealed is:

$$\begin{aligned}\theta - (\kappa - \beta_k + 1) &= (\kappa - \eta) - (\kappa - \beta_k + 1)\\ &= \beta_k - \eta - 1.\end{aligned}$$

If unrevealed commitments are all commitments of "0", then the losing bid is $\beta_k - 1$, whereas if they are all commitments of "1", then the losing bid is

$$\beta_l = (\beta_k - 1) - (\beta_k - \eta - 1) = \eta.$$

This means that a losing bid can be any value in the range $[\eta, \beta_k - 1]$, that is, while the losers do not reveal any extra information regarding their bids, the losing bids are less than the winning bid $\beta_k$. ∎

**Theorem 5.10** *The auction protocol VNR takes at most $O(\theta)$ rounds of communications and $O(\theta n^2 \log^2 n)$ computation cost where $\theta$ denotes the price range.*

The analysis is similar to the computation cost of the protocol VR except that here we have $\theta n$ polynomials $g_{ij}(x)$ of degree $n-1$ to be evaluated at $n$ points for $n$ bidders.

### 5.5.3 Efficient Verifiable Protocol with Non-Repudiation

We now modify our previous approach in order to construct a more efficient protocol that provides partial privacy of bids. Let $\lambda = \lceil \log_2 \theta \rceil$ where $\theta$ denotes our price range. This scheme is presented in Figures 5.7 and 5.8. We also provide an example of this scheme.

**Example 5.11** *Suppose each $\beta_i \in [0, 7]$ and all computations are performed in the field $\mathbb{Z}_{13}$. Assume $\beta_i = 7 - (101)_2 = 7 - 5 = 2$ and the winning price is $\beta_k = 5$ or $\beta_k = 3$ in two different scenarios. Here $\mathcal{M}_i(x) \in [0, 7)$ if $x = 0$; otherwise, $\mathcal{M}_i(x) \in [7, 13)$. Here, the binary representation $(1, 0, 1)$ has the corresponding mapping $(11, 5, 9)$.*

EVNR-a: Sealed-Bid First-Price Auction

1. **Initialize:** The trusted initializer $T$ first provides some private data through pair-wise channels and then leaves the scheme.

   (a) He randomly selects $\lambda$ polynomials $g_{i1}(x), g_{i2}(x), \ldots, g_{i\lambda}(x) \in \mathbb{Z}_q[x]$ of degree $n-1$ for each bidder $B_i$, and privately sends these polynomials to $B_i$ for $1 \le i \le n$.

   (b) He then selects $n-1$ distinct points $(\omega^{x^k_{ij}}, y^k_{ij})$ for $1 \le k \le n$ and $k \ne i$ uniformly at random on each $g_{ij}(x)$, where $1 \le j \le \lambda$. He finally sends these points to $B_k$. The following matrix shows the information that each $B_k$ receives, i.e., all entries in $k^{th}$ row:

$$\mathcal{E}_{n\times\lambda n} = \begin{pmatrix} g_{11}(x) & \ldots & g_{1\lambda}(x) & \ldots & (\omega^{x^1_{n1}}, y^1_{n1}) & \ldots & (\omega^{x^1_{n\lambda}}, y^1_{n\lambda}) \\ (\omega^{x^2_{11}}, y^2_{11}) & \ldots & (\omega^{x^2_{1\lambda}}, y^2_{1\lambda}) & \ldots & (\omega^{x^2_{n1}}, y^2_{n1}) & \ldots & (\omega^{x^2_{n\lambda}}, y^2_{n\lambda}) \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ (\omega^{x^n_{11}}, y^n_{11}) & \ldots & (\omega^{x^n_{1\lambda}}, y^n_{1\lambda}) & \ldots & g_{n1}(x) & \ldots & g_{n\lambda}(x) \end{pmatrix}.$$

2. **Start:** when the auction starts, there is a specific time interval in which bidders are allowed to commit to their bids.

   (a) Each $B_i$ first defines his bid $\beta_i$ as shown below, where $(b_{i\lambda} \ \ldots \ b_{i2} \ b_{i1})_2$ is the binary representation of $\kappa - \beta_i$ and $b_{ij} \in \{0,1\}$:

$$\beta_i = \kappa - \sum_{j=1}^{\lambda} b_{ij} 2^{j-1}. \tag{5.6}$$

   (b) Each bidder $B_i$ then applies a random mapping $\mathcal{M}_i(x) : \{0,1\} \to \mathbb{Z}_q$ to convert list $(b_{i\lambda}\ ,\ \ldots\ ,\ b_{i2}\ ,\ b_{i1})$ to a new list $(b'_{i\lambda}\ ,\ \ldots\ ,\ b'_{i2}\ ,\ b'_{i1})$ so that each $b'_{ij} \in \mathbb{Z}_q$. We require that $\mathcal{M}_i(x) \in [0, q/2)$ if $x = 0$; otherwise, $\mathcal{M}_i(x) \in [q/2, q)$.

   (c) Finally, each bidder $B_i$ for $1 \le i \le n$ commits to $b'_{ij}$ by $\alpha_{ij} = g_{ij}(\omega^{b'_{ij}})$ for $1 \le j \le \lambda$ and broadcasts all $\alpha_{ij}$ to the other bidders.

Figure 5.7: a. Efficient Verifiable Protocol with Non-Repudiation

EVNR-b: Sealed-Bid First-Price Auction

3. **Close:** after the closing time, bidders set the initial price $\gamma$ to be the highest possible price, i.e., $\gamma = \kappa$, and then define winners as follows:

   (a) $B_k$ who has committed to $\gamma$ claims he is the winner. Consequently, he must prove $\beta_k = \gamma$. Therefore, he reveals $g_{kj}(x)$ and $b'_{kj}$ for $1 \leq j \leq \lambda$. By using the inverse mappings $[0, q/2) \to 0$ and $[q/2, q) \to 1$, $b_{kj}$ for $1 \leq j \leq \lambda$ are recovered and $\beta_i$ is computed as follows:

   $$\beta_i = \kappa - \sum_{j=1}^{\lambda} b_{ij} 2^{j-1}. \tag{5.7}$$

   (b) If $\beta_k = \gamma$, the other bidders then investigate the validity of $\alpha_{kj} = g_{kj}(\omega^{b'_{kj}})$ for $1 \leq j \leq \lambda$. They also check to see if each set of $n-1$ points $(\omega^{x^i_{kj}}, y^i_{kj})$ for $1 \leq i \leq n$ and $i \neq k$ are on $g_{kj}(x)$'s. If these conditions hold, $B_k$ is accepted as the winner, otherwise, his claim is rejected.

   (c) Each loser $B_l$ must prove $\beta_l < \beta_k$. Therefore, $B_l$ reveals a minimal subset $\mathcal{J}$ of his commitments $b'_{lj}$ such that the following condition holds:

   $$\sum_{j \in \mathcal{J}} (b_{lj} \times 2^{j-1}) > \kappa - \beta_k. \tag{5.8}$$

   where $b_{lj}$ is the inverse image of $b'_{lj}$. Obviously, $B_l$ needs to provide valid proofs for the $b'_{lj}$'s. (The protocol still works even if $\mathcal{J}$ is not minimal. However, this would reveal more information about a loser's bid.)

   (d) If no one claims to be the winner or the bidder who claimed to be the winner could not prove his plea, then bidders decrease the selling price by one, i.e., $\gamma = \kappa - 1$, and the procedure from stage (a) is repeated.

Figure 5.8: b. Efficient Verifiable Protocol with Non-Repudiation

- *Suppose the winning bid is $\beta_k = 5$. The loser $B_i$ reveals his $3^{rd}$ commitment to prove $(1 \times 2^2) > 7 - 5$. This shows his bid is at most $7 - 4 = 3$, which is less than the winning price.*

- *Suppose the winning bid is $\beta_k = 3$. The loser $B_i$ reveals his $3^{rd}$ and $1^{st}$ commitments to prove $(1 \times 2^2 + 1 \times 2^0) > 7 - 3$. This shows his bid is at most $7 - 5 = 2$, which is less than the winning price.*

**Theorem 5.12** *The proposed auction protocol EVNR, presented in Figures 5.7 and 5.8, defines auction outcomes correctly under the honest majority assumption with a small probability of error, as computed in Theorem 5.3. This protocol partially protects the losing bids and satisfies the non-repudiation property.*

**Proof.** Similar to the previous theorem, we analyze part $(c)$ of stage 3. Here we show a partial information leakage that occurs. As presented in Figure 5.7, each bidder $B_i$ commits to $\lambda$ values, where $\lambda = \lceil \log_2 \theta \rceil$. According to Equation (5.8), a loser $B_l$ reveals $|\mathcal{J}|$ commitments of 1's in part $(c)$ of stage 3. This has two meanings:

1. Up to this time of the auction, the revealed portion of his bid is:
$$\beta_l = \kappa - \sum_{j \in \mathcal{J}} (b_{lj} \times 2^{j-1}) \quad \text{by (5.6) and (5.8)},$$
where $\beta_l < \beta_k$.

2. The number of commitments that have not been yet revealed is:
$$\lambda - |\mathcal{J}| = |\mathcal{J}'|.$$

where $\mathcal{J}'$ is the complement of $\mathcal{J}$. If unrevealed commitments are all commitments of "0", then the losing bid is $\kappa - \sum_{j \in \mathcal{J}} (b_{lj} \times 2^{j-1})$, whereas if they are all commitments of "1", then the losing bid is:

$$\begin{aligned}
\beta_l &= \left(\kappa - \sum_{j \in \mathcal{J}} (b_{lj} \times 2^{j-1})\right) - \sum_{j \in \mathcal{J}'} (b_{lj} \times 2^{j-1}) \\
&= \kappa - \sum_{j=1}^{\lambda} (b_{lj} \times 2^{j-1}) \quad \text{where } b_{lj} = 1 \\
&= \kappa - \theta \\
&= \eta.
\end{aligned}$$

This means that a losing bid can be some values in the range $[\eta, \beta_l]$, since only certain bit combinations are possible. That is, the losers reveal upper bounds of their bids, and also they prove that the losing bids are less than $\beta_k$. ∎

**Theorem 5.13** *The auction protocol EVNR takes at most $O(\theta)$ rounds of communications and $O(\lambda n^2 \log^2 n) = O(\log_2 \theta \times n^2 \log^2 n)$ computation cost where $\theta$ denotes the price range.*

The analysis is similar to the computation cost of the protocol VNR, except that here we have $\lambda n$ polynomials $g_{ij}(x)$ of degree $n-1$ where $\lambda = \lceil \log_2 \theta \rceil$.

## 5.6 Conclusion

We initially illustrated the lack of unconditional security in sealed-bid auction protocols, and then proposed three unconditionally secure schemes. We constructed a multicomponent commitment scheme MCS and proposed three secure first-price auction protocols base on that construction. Table 5.1 represents outlines of our contributions.

| | **Assumption** | **Rounds** | **Cost** | **Private** | **Verify** | **Non-Rep** |
|---|---|---|---|---|---|---|
| VR | honest | | $O(n^2 \log^2 n)$ | yes | yes | no |
| VNR | majority | $O(\theta)$ | $O(\theta n^2 \log^2 n)$ | yes | yes | yes |
| EVNR | assumption | | $O(n^2 \log_2 \theta \log^2 n)$ | partial | yes | yes |

Table 5.1: Unconditionally Secure First-Price Auction Protocols Using MCS

We should note that the last two protocols are still secure without using the random mapping $\mathcal{M}_i(x) : \{0,1\} \rightarrow \mathbb{Z}_q$, as shown in Figures 5.5 and 5.7; step $2(b)$. In other words, bidders $B_i$ can simply commit to "0" or "1" by using $g_{ij}(x)$. In this case, committing to "0" means commitment only to the constant term of $g_{ij}(x)$, but we prefer to use the entire coefficients of $g_{ij}(x)$ for each commitment. This provides a better protection from a system-level perspective, where the attacker might be able to read one word of the bidder's memory but not lots of words. However, this is outside the scope of our threat model.

Our unconditionally secure sealed-bid auction protocols have *full privacy*, that is, $(n-1)$ participants cannot learn a committed value. The honest majority assumption (i.e., less than $n/2$ players are compromised by an active adversary) guarantees the *correctness* of the protocols.

# Chapter 6

# Conclusion and Future Directions

In this thesis, four new cryptographic primitives were introduced, that is, *social secret sharing, socio-rational secret sharing, dynamic secret sharing, and multicomponent commitment*. Our proposed schemes were constructed in different models ranging from "social" to "adversarial" settings. Moreover, each scheme was motivated with a new application, for instance, *self-organizing clouds, repeated sealed-bid auctions, sequential secret sharing,* and *first-price sealed-bid auctions*.

Although cryptography has a large and diverse impact, there still exist many untouched areas that need to be explored, specially from a multidisciplinary perspective. Moreover, innovative applications can be developed through constructions of novel cryptographic building blocks. Therefore, it is constructive to approach and modify the existing primitives from diverse angles for new development.

## 6.1 Future Extensions

In the context of social secret sharing, we would like to perform experiments on various trust functions and also consider other reputation management systems. For instance, schemes that use a *referral chain*, where two players who are interacting for the first time can gain some information with respect to each other's reputation through other parties or common friends. In addition, we are interested in deploying such a construction over the cloud for an experimental analysis.

In the context of socio-rational secret sharing, we are interested in working on more complicated models. For instance, we would like to scrutinize the impact of a situation in which a player is involved in various social games while he is holding different reputation values associated with each society. It would be also interesting to construct a *hybrid model* in which both reputation and belief are considered. In this case, reputation can be seen as a consequence of the past behavior whereas belief can be viewed as an anticipation of the future behavior.

In the context of dynamic secret sharing, we are interested in exploring other techniques for threshold modification and a player's enrollment. We also would like to extend our sequential secret sharing scheme to an active adversary setting. As a final extension, we intend to propose new protocols for second-price and combinatorial sealed-bid auctions.

## 6.2 Future Research Agenda

The growth of e-technology and online services has created a remarkable opportunity for secure multiparty computation. As our future agenda, we would like to propose new applications of multiparty computation in the context of "financial engineering". Indeed, the recent economic downturn raised many concerns in research communities. Scientists would like to scrutinize this phenomenon by analyzing the existing computational models of financial schemes. For this reason, our future research intends to explore potential solutions in this direction through a new cryptographic infrastructure.

We intend to extend the current computational model of multiparty computation to a more general financial paradigm, named *securely computable economic model*. It is worth mentioning that "sealed-bid auctions" are just an instance of these kinds of financial functionalities. Our intention is to study various economic frameworks to introduce a novel financial paradigm where parameters and computations are injected in and modeled by secure multiparty computation protocols. In fact, "lack of transparency" and consequently "economic frauds" are believed to be the major reasons for such economic crashes. This problem can be approached by constructing a new computational infrastructure, in which information leakage is minimized while enforcing transparency.

As a strategic decision-making scenario, consider a situation in which a private sector $\mathcal{P}$ wants to get financial support from an investment institution $\mathcal{I}$. The intention of $\mathcal{P}$ is to keep its private information secure from other competitors in the market. On the other hand, before approving the investment, $\mathcal{I}$ would like to make sure that:

(a) Certain constraints on savings, budgets, and cash flows of $\mathcal{P}$ are satisfied.

(b) Information related to ongoing secure negotiations with potential buyers is truthful.

Other parties such as banks or the tax bureau, or governmental agencies, might be involved in this mutual agreement. It is not hard to show that the agreement functionality can be modeled as a privacy-preserving computation with transparency enforcement so that any inconsistency can be detected.

We first have to model the proposed paradigm in a theoretical setting. For each problem instance, we would like to analyze the constructed model from various aspects, e.g., efficiency and complexity. Our objective is to focus on real-world problem instances such as *investment agreement, stock exchange, strategic negotiations*, etc. We do believe that this research agenda provides a better understanding of cryptographic constructions, and it expands the application of theoretical cryptography to other computational fields such as financial engineering.

# References

[1] Masayuki Abe and Koutarou Suzuki. M+1-st price auction using homomorphic encryption. In *5th International Workshop on Practice and Theory in Public Key Cryptography, PKC'02*, volume 2274 of *LNCS*, pages 115–124. Springer, 2002.

[2] Ittai Abraham, Danny Dolev, Rica Gonen, and Joseph Y. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In *25th Annual ACM Symposium on Principles of Distributed Computing, PODC'06*, pages 53–62, 2006.

[3] Gilad Asharov, Ran Canetti, and Carmit Hazay. Towards a game theoretic view of secure computation. In *30th Annual International Conference on the Theory and Applications of Cryptographic Techniques EUROCRYPT*, volume 6632 of *LNCS*, pages 426–445. Springer, 2011.

[4] Gilad Asharov and Yehuda Lindell. Utility dependence in correct and fair rational secret sharing. In *29th Annual International Cryptology Conference, CRYPTO'09*, volume 5677 of *LNCS*, pages 559–576. Springer, 2009.

[5] Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In *8th Annual ACM Symposium on Principles of Distributed Computing, PODC*, pages 201–209, 1989.

[6] Susan G. Barwick, Wen Ai Jackson, and Keith M. Martin. Updating the parameters of a threshold scheme by minimal broadcast. *IEEE Transactions on Information Theory*, 51(2):620–633, 2005.

[7] Olivier Baudron and Jacques Stern. Non-interactive private auctions. In *5th International Conference on Financial Cryptography, FC'01*, volume 2339 of *LNCS*, page 364377. Springer, 2001.

[8] Donald Beaver. Multiparty protocols tolerating half faulty processors. In *9th Annual International Cryptology Conference, CRYPTO'89*, volume 435 of *LNCS*, pages 560–572. Springer, 1989.

[9] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *20th Annual ACM Symposium on Theory of Computing, STOC'88*, pages 1–10, 1988.

[10] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *8th Annual International Cryptology Conference, CRYPTO'88*, volume 403 of *LNCS*, pages 27–35. Springer, 1988.

[11] G. R. Blakley. Safeguarding cryptographic keys. In *National Computer Conference, NCC'79*, pages 313–317. AFIPS Press, 1979.

[12] Manuel Blum. Coin flipping by telephone: a protocol for solving impossible problems. *SIGACT News: a special issue on cryptography*, 15:23–27, 1983.

[13] C. Blundo, B. Masucci, DR Stinson, and R. Wei. Constructions and bounds for unconditionally secure non-interactive commitment schemes. *Designs, Codes and Cryptography*, 26(1):97–110, 2002.

[14] Carlo Blundo, Antonella Cresti, Alfredo De Santis, and Ugo Vaccaro. Fully dynamic secret sharing schemes. *Theoretical Computer Science*, 165(2):407–440, 1996.

[15] Dan Boneh and Matthew Franklin. Efficient generation of shared RSA keys. *Journal of ACM*, 48(4):702–722, 2001.

[16] F. Brandt. A verifiable, bidder-resolved auction protocol. In *5th International Workshop on Deception, Fraud and Trust in Agent Societies, Special Track on Privacy and Protection with Multi-Agent Systems*, pages 18–25, 2002.

[17] Felix Brandt. Cryptographic protocols for secure second-price auctions. In *5th International Workshop on Cooperative Information Agents, CIA'01*, volume 2182 of *LNCS*, pages 154–165. Springer, 2001.

[18] Felix Brandt. How to obtain full privacy in auctions. *International Journal of Information Security*, 5(4):201–216, 2006.

[19] Felix Brandt and Tuomas Sandholm. (Im)possibility of unconditionally privacy-preserving auctions. In *3rd ACM International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS'04*, volume 2, pages 810–817, 2004.

[20] Felix Brandt and Tuomas Sandholm. On the existence of unconditionally privacy-preserving auction protocols. *ACM Transactions on Information and System Security*, 11(2):1–21, 2008.

[21] James Broberg, Srikumar Venugopal, and Rajkumar Buyya. Market-oriented grids and utility computing: The state-of-the-art and future directions. *Journal of Grid Computing*, 6(3):255–276, 2008.

[22] Christian Cachin. Efficient private bidding and auctions with an oblivious third party. In *6th ACM Conference on Computer and Communications Security, CCS'99*, pages 120–127, 1999.

[23] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In *20th Annual ACM Symposium on Theory of Computing, STOC'88*, pages 11–19, 1988.

[24] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *26th Annual IEEE Symposium on Foundations of Computer Science, FOCS'85*, pages 383–395, 1985.

[25] Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *3rd Theory of Cryptography Conference, TCC'06*, volume 3876 of *LNCS*, pages 285–304. Springer, 2006.

[26] Paolo D'Arco and Douglas R. Stinson. On unconditionally secure robust distributed key distribution centers. In *8th International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT'02*, volume 2501 of *LNCS*, pages 346–363. Springer, 2002.

[27] Yvo Desmedt and Sushil Jajodia. Redistributing secret shares to new access structures and its applications. Technical report, George Mason University, 1997.

[28] Yevgeniy Dodis, Shai Halevi, and Tal Rabin. A cryptographic solution to a game theoretic problem. In *20th Annual International Cryptology Conference, CRYPTO'00*, volume 1880 of *LNCS*, pages 112–130. Springer, 2000.

[29] Wenliang Du and Mikhail J. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *Workshop on New Security Paradigms, NSPW'01*, pages 13–22. ACM, 2001.

[30] Y. Frankel, P. Gemmell, P. D. MacKenzie, and Moti Yung. Optimal-resilience proactive public-key cryptosystems. In *38th Annual IEEE Symposium on Foundations of Computer Science, FOCS'97*, pages 384–393, 1997.

[31] Matthew K. Franklin and Michael K. Reiter. The design and implementation of a secure auction service. *IEEE Transactions on Software Engineering*, 22(5):302–312, 1996.

[32] Georg Fuchsbauer, Jonathan Katz, and David Naccache. Efficient rational secret sharing in standard communication networks. In *7th Theory of Cryptography Conference, TCC'10*, volume 5978 of *LNCS*, pages 419–436. Springer, 2010.

[33] Juan A. Garay, Berry Schoenmakers, and José Villegas. Practical and secure solutions for integer comparison. In *10th International Conference on Practice and Theory in Public-Key Cryptography, PKC'07*, volume 4450 of *LNCS*, pages 330–342. Springer, 2007.

[34] Joachim Von Zur Gathen and Jurgen Gerhard. *Modern Computer Algebra.* Cambridge University Press, New York, USA, 2003.

[35] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *33th Annual ACM Symposium on Theory of Computing, STOC'01*, pages 580–589, 2001.

[36] Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *17th annual ACM symposium on Principles of Distributed Computing, PODC'98*, pages 101–111, 1998.

[37] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *19th Annual ACM Symposium on Theory of Computing, STOC'87*, pages 218–229, 1987.

[38] Shafi Goldwasser. Multi party computations: past and present. In *16th Annual ACM Symposium on Principles of Distributed Computing, PODC'97*, pages 1–6, 1997.

[39] S. Dov Gordon and Jonathan Katz. Rational secret sharing, revisited. In *5th International Conference on Security and Cryptography for Networks, SCN'06*, volume 4116 of *LNCS*, pages 229–241. Springer, 2006.

[40] Ronen Gradwohl, Noam Livne, and Alon Rosen. Sequential rationality in cryptographic protocols. In *51th Annual IEEE Symposium on Foundations of Computer Science FOCS*, pages 623–632, 2010.

[41] Joseph Y. Halpern and Vanessa Teague. Rational secret sharing and multiparty computation: extended abstract. In *36th Annual ACM Symposium on Theory of Computing, STOC'04*, pages 623–632, 2004.

[42] Michael Harkavy, J. D. Tygar, and Hiroaki Kikuchi. Electronic auctions with private bids. In *3rd Conference on USENIX Workshop on Electronic Commerce, WOEC'98*, pages 61–74. USENIX Association, 1998.

[43] J. He and E. Dawson. Multistage secret sharing based on one-way function. *Electronics Letters*, 30(19):1591–1592, 1994.

[44] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *15th Annual International Cryptology Conference, CRYPTO'95*, volume 963 of *LNCS*, pages 339–352. Springer, 1995.

[45] Sergei Izmalkov, Silvio Micali, and Matt Lepinski. Rational secure computation and ideal mechanism design. In *46th Annual IEEE Symposium on Foundations of Computer Science, FOCS'05*, pages 585–595, 2005.

[46] Markus Jakobsson and Ari Juels. Mix and match: Secure function evaluation via ciphertexts. In *6th International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT'00*, volume 1976 of *LNCS*, pages 162–177. Springer, 2000.

[47] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

[48] Jonathan Katz. Bridging game theory and cryptography: Recent results and future directions. In *5th Theory of Cryptography Conference, TCC'08*, volume 4948 of *LNCS*, pages 251–272. Springer, 2008.

[49] Hiroaki Kikuchi. (m+1)st-price auction protocol. In *5th International Conference on Financial Cryptography, FC'01*, volume 2339 of *LNCS*, pages 351–363. Springer, 2001.

[50] Hiroaki Kikuchi, Michael Harkavy, and J. D. Tygar. Multi-round anonymous auction protocols. *IEICE Transaction on Information and Systems*, 82:769–777, 1999.

[51] Hiroaki Kikuchi, Shinji Hotta, Kensuke Abe, and Shohachiro Nakanishi. Distributed auction servers resolving winner and winning bid without revealing privacy of bids. In *7th IEEE International Conference on Parallel and Distributed Systems, ICPADS'00*, pages 307–312, 2000.

[52] Gillat Kol and Moni Naor. Cryptography and game theory: Designing protocols for exchanging information. In *5th Theory of Cryptography Conference, TCC'08*, volume 4948 of *LNCS*, pages 320–339. Springer, 2008.

[53] Gillat Kol and Moni Naor. Games for exchanging information. In *40th Annual ACM Symposium on Theory of Computing, STOC'08*, pages 423–432, 2008.

[54] Matt Lepinski, Silvio Micali, Chris Peikert, and Abhi Shelat. Completely fair sfe and coalition-safe cheap talk. In *23th Annual ACM Symposium on Principles of Distributed Computing, PODC'04*, pages 1–10, 2004.

[55] Matt Lepinski, Silvio Micali, and Abhi Shelat. Collusion-free protocols. In *37th Annual ACM Symposium on Theory of Computing, STOC'05*, pages 543–552, 2005.

[56] Helger Lipmaa, N. Asokan, and Valtteri Niemi. Secure Vickrey auctions without threshold trust. In *6th International Conference on Financial Cryptography, FC'02*, volume 2357 of *LNCS*, pages 87–101. Springer, 2002.

[57] Anna Lysyanskaya and Nikos Triandopoulos. Rationality and adversarial behavior in multi-party computation. In *26th International Cryptology Conference, CRYPTO'06*, volume 4117 of *LNCS*, pages 180–197. Springer, 2006.

[58] F.J. MacWilliams and N.J.A. Sloane. *The theory of error-correcting codes.* North-Holland Amsterdam, 1978.

[59] Ayako Maeda, Atsuko Miyaji, and Mitsuru Tada. Efficient and unconditionally secure verifiable threshold changeable scheme. In *6th Australasian Conference Information Security and Privacy, ACISP'01*, LNCS, pages 403–416. Springer, 2001.

[60] G.J. Mailath and L. Samuelson. *Repeated games and reputations: long-run relationships.* Oxford University Press, USA, 2006.

[61] Shaik Maleka, Amjed Shareef, and C. Pandu Rangan. Rational secret sharing with repeated games. In *4th International Conference on Information Security Practice and Experience, ISPEC'08*, volume 4991 of *LNCS*, pages 334–346. Springer, 2008.

[62] Keith M. Martin, Josef Pieprzyk, Reihaneh Safavi-Naini, and Huaxiong Wang. Changing thresholds in the absence of secure channels. In *4th Australasian Conference Information Security and Privacy, ACISP'99*, volume 1587 of *LNCS*, pages 177–191. Springer, 1999.

[63] Keith M. Martin, Reihaneh Safavi-Naini, and Huaxiong Wang. Bounds and techniques for efficient redistribution of secret shares to new access structures. *The Computer Journal*, 42(8):638–649, 1999.

[64] Silvio Micali and Abhi Shelat. Purely rational secret sharing. In *6th Theory of Cryptography Conference, TCC'09*, volume 5444 of *LNCS*, pages 54–71. Springer, 2009.

[65] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. A computational model of trust and reputation. In *35th Annual Hawaii International Conference on System Sciences, ICSS'02*, pages 2431–2439, 2002.

[66] Khanh Quoc Nguyen and Jacques Traoré. An online public auction protocol protecting bidder privacy. In *5th Australasian Conference on Information Security and Privacy, ACISP'00*, volume 1841 of *LNCS*, pages 427–442. Springer, 2000.

[67] Ventzislav Nikov and Svetla Nikova. On proactive secret sharing schemes. In *11th International Workshop on Selected Areas in Cryptography, SAC'04*, volume 3357 of *LNCS*, pages 308–325. Springer, 2004.

[68] Takashi Nishide and Kazuo Ohta. Constant-round multiparty computation for interval test, equality test, and comparison. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 90-A(5):960–968, 2007.

[69] Takashi Nishide and Kazuo Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In *10th International Conference on Practice and Theory in Public-Key Cryptography, PKC'07*, volume 4450 of *LNCS*, pages 343–360. Springer, 2007.

[70] Mehrdad Nojoumian and Timothy C. Lethbridge. A new approach for the trust calculation in social networks. *E-business and Telecommunication Networks: 3rd International Conference on E-Business ICE-B, Selected Papers*, 9:64–77, 2008.

[71] Mehrdad Nojoumian and Douglas R. Stinson. Brief announcement: secret sharing based on the social behaviors of players. In *29th ACM symposium on Principles of distributed computing, PODC'10*, pages 239–240, 2010.

[72] Mehrdad Nojoumian and Douglas R. Stinson. Unconditionally secure first-price auction protocols using a multicomponent commitment scheme. In *12th International Conference on Information and Communications Security, ICICS'10*, volume 6476 of *LNCS*, pages 266–280. Springer, 2010.

[73] Mehrdad Nojoumian and Douglas R. Stinson. On dynamic threshold schemes and sequential secret sharing. In *submitted to the Advances in Mathematics of Communications*, 2012.

[74] Mehrdad Nojoumian and Douglas R. Stinson. Social secret sharing in cloud computing using a new trust function. In *to appear in 10th Annual Conference on Privacy, Security and Trust, PST'12*. IEEE, 2012.

[75] Mehrdad Nojoumian and Douglas R. Stinson. Socio-rational secret sharing as a new direction in rational cryptography. In *submitted to the 3th Conference on Decision and Game Theory for Security, GameSec'12*, 2012.

[76] Mehrdad Nojoumian, Douglas R. Stinson, and Morgan Grainger. Unconditionally secure social secret sharing scheme. *IET Information Security, Special Issue on Multi-Agent and Distributed Information Security*, 4(4):202–211, 2010.

[77] Kazumasa Omote and Atsuko Miyaji. A second-price sealed-bid auction with the discriminant of the $p_0$-th root. In *6th International Conference on Financial Cryptography, FC'02*, volume 2357 of *LNCS*, pages 57–71. Springer, 2002.

[78] Shien Jin Ong, David C. Parkes, Alon Rosen, and Salil P. Vadhan. Fairness with an honest minority and a rational majority. In *6th Theory of Cryptography Conference, TCC'09*, volume 5444 of *LNCS*, pages 36–53. Springer, 2009.

[79] Martin J. Osborne and A. Rubinstein. *A course in game theory*. MIT press, 1994.

[80] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks: extended abstract. In *10th Annual ACM Symposium on Principles of Distributed Computing, PODC'91*, pages 51–59, 1991.

[81] David C. Parkes, Michael O. Rabin, and Christopher Thorpe. Cryptographic combinatorial clock-proxy auctions. In *13th International Conference on Financial Cryptography, FC'09*, volume 5628 of *LNCS*, pages 305–324. Springer, 2009.

[82] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *11th Annual International Cryptology Conference, CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, 1991.

[83] Kun Peng, Colin Boyd, and Ed Dawson. Optimization of electronic first-bid sealed-bid auction based on homomorphic secret sharing. In *1st International Conference on Cryptology in Malaysia, Mycrypt'05*, volume 3715 of *LNCS*, pages 84–98. Springer, 2005.

[84] Kun Peng, Colin Boyd, Ed Dawson, and Kapali Viswanathan. Robust, privacy protecting and publicly verifiable sealed-bid auction. In *4th International Conference on Information and Communications Security, ICICS'02*, LNCS, pages 147–159. Springer, 2002.

[85] Kun Peng, Colin Boyd, Ed Dawson, and Kapali Viswanathan. Five sealed-bid auction models. In *the Australasian Information Security Workshop Conference, AISW'03*, volume 21, pages 77–86. Australian Computer Society, 2003.

[86] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *21st Annual ACM Symposium on Theory of Computing, STOC'89*, pages 73–85, 1989.

[87] Rolf S. Rees, Douglas R. Stinson, Ruizhong Wei, and G. H. John van Rees. An application of covering designs: determining the maximum consistent set of shares in a threshold scheme. *Ars Combinatoria*, 53:225–237, 1999.

[88] Ronald L. Rivest. Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer. Technical report, Massachusetts Institute of Technology, 1999.

[89] Michael H Rothkopf, Thomas J Teisberg, and Edward P Kahn. Why are Vickrey auctions rare? *Journal of Political Economy*, 98(1):94–109, 1990.

[90] Kazue Sako. An auction protocol which hides bids of losers. In *3rd International Workshop on Practice and Theory in Public Key Cryptography, PKC'00*, volume 1751 of *LNCS*, pages 422–432. Springer, 2000.

[91] K. Sakurai and S. Miyazaki. A bulletin-board based digital auction scheme with bidding down strategy. In *the international workshop on cryptographic techniques and E-commerce, CrypTEC'99*, pages 180–187, 1999.

[92] Tuomas Sandholm. Issues in computational Vickrey auctions. *International Journal of Electronic Commerce*, 4(3):107–129, 2000.

[93] Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi. Efficient node admission for short-lived mobile ad hoc networks. In *13th IEEE International Conference on Network Protocols, ICNP'05*, pages 269–278, 2005.

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

[95] Gustavus J. Simmons. How to (really) share a secret. In *8th Annual International Cryptology Conference, CRYPTO'88*, volume 403 of *LNCS*, pages 390–448. Springer, 1988.

[96] Ron Steinfeld, Huaxiong Wang, and Josef Pieprzyk. Lattice-based threshold-changeability for standard Shamir secret-sharing schemes. In *10th International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT'04*, volume 3329 of *LNCS*, pages 170–186. Springer, 2004.

[97] Douglas R. Stinson. *Cryptography: Theory and Practice, 3rd Edition.* CRC Press, 2005.

[98] Douglas R. Stinson and Ruizhong Wei. Unconditionally secure proactive secret sharing scheme with combinatorial structures. In *6th Annual International Workshop on Selected Areas in Cryptography, SAC'99*, volume 1758 of *LNCS*, pages 200–214. Springer, 1999.

[99] Koutarou Suzuki, Kunio Kobayashi, and Hikaru Morita. Efficient sealed-bid auction using hash chain. In *3rd Annual International Conference on Information Security and Cryptology, ICISC'00*, volume 2015 of *LNCS*, pages 183–191. Springer, 2000.

[100] Koutarou Suzuki and Makoto Yokoo. Secure combinatorial auctions by dynamic programming with polynomial secret sharing. In *6th International Conference on Financial Cryptography, FC'02*, volume 2357 of *LNCS*, pages 44–56. Springer, 2002.

[101] Koutarou Suzuki and Makoto Yokoo. Secure multi-attribute procurement auction. In *6th International Workshop on Information Security Applications, WISA'05*, volume 3786 of *LNCS*, pages 306–317. Springer, 2005.

[102] Christophe Tartary and Huaxiong Wang. Dynamic threshold and cheater resistance for Shamir secret sharing scheme. In *2nd SKLOIS Conference on Information Security and Cryptology, Inscrypt'06*, volume 4318 of *LNCS*, pages 103–117. Springer, 2006.

[103] Tamir Tassa. Hierarchical threshold secret sharing. In *1st Theory of Cryptography Conference, TCC'04*, volume 2951 of *LNCS*, pages 473–490. Springer, 2004.

[104] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1):8–37, 1961.

[105] Kapali Viswanathan, Colin Boyd, and Ed Dawson. A three phased schema for sealed bid auction system design. In *Proceedings of the 5th Australasian Conference on Information Security and Privacy, ACISP'00*, volume 1841 of *LNCS*, pages 412–426. Springer, 2000.

[106] William Voorsluys, James Broberg, and Rajkumar Buyya. *Cloud Computing: Principles and Paradigms*. John Wiley and Sons, 2011.

[107] Yuji Watanabe and Hideki Imai. Reducing the round complexity of a sealed-bid auction protocol with an off-line ttp. In *7th ACM Conference on Computer and Communications Security, CCS'00*, pages 80–86, 2000.

[108] Andrew C. Yao. Protocols for secure computations. In *23rd Annual IEEE Symposium on Foundations of Computer Science, FOCS'82*, pages 160–164, 1982.

[109] Makoto Yokoo and Koutarou Suzuki. Secure multi-agent dynamic programming based on homomorphic encryption and its application to combinatorial auctions. In *1st ACM International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS'02*, pages 112–119, 2002.

[110] Makoto Yokoo and Koutarou Suzuki. Secure generalized Vickrey auction without third-party servers. In *8th International Conference on Financial Cryptography, FC'04*, volume 3110 of *LNCS*, pages 132–146. Springer, 2004.

[111] Bin Yu and Munindar P. Singh. A social mechanism of reputation management in electronic communities. In *4th International Workshop on Cooperative Information Agents, CIA'00*, volume 1860 of *LNCS*, pages 154–165. Springer, 2000.

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