Social Secret Sharing in Cloud Computing Using a New Trust Function

Mehrdad Nojoumian and Douglas R. Stinson
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada
mnojoumi@cs.uwaterloo.ca, dstinson@math.uwaterloo.ca

***Abstract*—We first review the notion of social secret sharing and its trust function. We then illustrate how this construction can be used in cloud computing to create a self-organizing environment. In fact, we show distributed secure systems using threshold secret sharing can be adjusted automatically based on the resource availability of the cloud providers. Accordingly, we propose a new trust function with social characteristics in order to improve the existing social secret sharing scheme.**

Keywords: trust modeling, secret sharing, cloud computing.

I. Introduction

In a *secret sharing* scheme, a secret ξ is divided into n shares in order to be distributed among a set of players. Subsequently, an authorized subset of players collaborate to reconstruct the secret [1] [2]. In particular, a (t, n)-*threshold secret sharing* (*TSS*) scheme consists of two phases: *sharing* and *recovery*. All computations are performed in a finite field $\mathbb{Z}_q$ where q is a prime number:

1) **Secret Sharing**: a dealer selects $f(x) \in \mathbb{Z}_q[x]$ of degree $t-1$ such that $f(0) = \xi$ is the secret. The dealer then sends shares $f(i)$ to P_i for $1 \leq i \leq n$, that is, different points on the secret sharing polynomial $f(x)$. He finally leaves the scheme.
2) **Secret Recovery**: subsequently, any subset Δ of at least t players can collaborate to reconstruct the secret by Lagrange interpolation in the absence of the dealer:

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

Any t players can combine their shares to reveal the secret but no set of $t-1$ parties can learn the secret. An example of this scheme is provided in Appendix VI-A. In fact, secret sharing is one of most important primitives used in different cryptographic constructions, for instance, in *secure multiparty computation* (*MPC*) where various players cooperate to jointly compute a function based on the secret data they each provide. As an example, we can refer to first-price sealed-bid auctions in which the maximum bid is determined while the losing bids are kept private [3].

For further technical discussions, we first review various types of adversarial models. We then recall three kinds of secret sharing schemes.

- *Passive versus Active Adversary*: in the former case, players follow the protocols correctly but they are curious to learn the secret (a.k.a honest-but-curious adversary). In the latter case, players may deviate from the protocols (e.g., to prevent the secret recovery or reconstruct an incorrect secret) while trying to learn the secret.
- *Static versus Mobile Adversary*: the former refers to an adversary who corrupts the players ahead of time whereas the latter refers to an adversary who corrupts the players while the protocol is executing.
- *Computational versus Unconditional Security*: in the former case, security of the protocols relies on computational assumptions (for instance, the hardness of factoring or discrete logarithm), whereas in the latter case, the adversary has unlimited computational power.

Next, we show how various secret sharing schemes are designed to deal with different adversarial models.

In a *verifiable secret sharing* (*VSS*) scheme [4], players can verify the consistency of the shares in both sharing and recovery phases. The authors in [5] [6] provide the first unconditionally secure VSS when $t < \frac{n}{3}$ with a zero probability of error. In this setting, each pair of the players are connected with a secure private channel. To tolerate a higher threshold $t < \frac{n}{2}$, the authors in [7] [8] assume the existence of both private channels and a broadcast channel. This protocol has a negligible probability of error. To simplify these constructions, the authors in [9] [10] propose verifiable schemes based on symmetric bivariate polynomials in an unconditionally secure setting. These protocols also assume the existence of private channels and a broadcast channel where $t < \frac{n}{4}$.

The notion of *proactive secret sharing* (*PSS*) is proposed in [11], where the shares of the players are updated without changing the secret. This can be done by adding the shares of a new polynomial g with zero constant term to the shares of the original secret sharing polynomial f with constant term ξ. As a result, the new secret sharing polynomial will be $\hat{f} = f + g$ where $\hat{f}(0) = \xi$. In other words, players frequently change the secret sharing polynomial to deal with a mobile adversary [12] who can incrementally collect the shares of the players while the protocol is executing. An example of proactive secret sharing is provided in Appendix VI-B.

To assign multiple shares rather than a single share to some players, *weighted secret sharing* (*WSS*) is introduced in [13]. Suppose in a company, the secret key of the safe deposit box is shared among chief executive officer, director, and two managers. Let assume these parties receive $4, 3, 2, 2$ shares respectively on a polynomial of degree 5. As a result, the chief executive officer can open the safe deposit box with the director or one manager but the other parties can only open it if they all collaborate. Indeed, weighted secret sharing is used to prioritize different players in a hierarchy structure.

A. Motivation and Contribution

We illustrate how *social secret sharing* (*SSS*) can be applied in distributed secure systems using cloud computing infrastructures. Moreover, we intend to improve social secret sharing by proposing a new trust function.

Therefore, as our contributions, 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, we provide a new trust function with social properties in order to improve the existing social secret sharing scheme.

B. Organization

The rest of this paper is organized as follows. Section II reviews the notion of social secret sharing. Section III illustrates a new application of this scheme in cloud computing. Section IV proposes a new trust function. Finally, Section V provides concluding remarks.

II. Review of Social Secret Sharing

We now review social secret sharing introduced by Nojoumian et al. [14] [15]. In this construction, the number of shares allocated to each player depends on the player's reputation and the way he interacts with other parties. In other words, weights of the players are adjusted such that cooperative participants receive more shares compared to non-cooperative parties. This is similar to our social life where we share more secrets with those whom we fully trust and vice versa. Here is the definition of social secret sharing:

Definition 1: Social secret sharing is a three-tuple denoted as $(\mathcal{S}ha, \mathcal{T}un, \mathcal{R}ec)$ *consisting of secret sharing, social tuning, and secret recovery respectively. The only difference compared to threshold secret sharing is the second stage, in which the weight of each player* P_i *is adjusted according to his reputation.*

A. Assumptions

The following properties are required to construct a social secret sharing scheme with secret ξ:

1) To recover secret ξ, the total weight of authorized players $P_i \in \Delta$ must be equal or greater than the threshold:

$$\sum_{P_i \in \Delta} w_i \geq t$$

where Δ denotes the set of uncorrupted participants. We later show that this set 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 ∇ denotes the set of corrupted players:

$$\sum_{P_i \in \nabla} w_i < t.$$

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

B. Social Tuning (weights adjustment)

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 for the cooperative player. This procedure is done in the absence of the dealer. For details, see the *enrollment* protocol in [14].
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 $\hat{f}(x)$, that share remains on the old secret sharing polynomial $f(x)$, as a result, the share will not be valid anymore.

In the next section, we recall the trust function that is used in social secret sharing for reputation measurement.

C. Trust Function

To measure the reputation of each player in a social secret sharing scheme, the authors use the trust calculation method proposed in [16]; this research significantly improves the well-known solution in [17]. We start with the following definition:

Definition 2: Let $\mathcal{T}_i^j(p)$ *be the trust value assigned by* P_j *to* P_i *in period* p*. Let* $\mathcal{T}_i : \mathbb{N} \mapsto \mathbb{R}$ *be the* **trust function** *representing 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 trust values (personal quantity) in order to compute a player's reputation (social quantity) [14].

For instance, let the trust values of P_1, P_2, P_3 with respect to P_4 be $\mathcal{T}_4^1(p) = 0.4$, $\mathcal{T}_4^2(p) = 0.5$, $\mathcal{T}_4^3(p) = 0.6$ accordingly. As a result, reputation of P_4 will be $\mathcal{T}_4(p) = 0.5$. In this paper, only a public value $\mathcal{T}_i(p)$ is assigned to each player P_i that represents his reputation, i.e., $\mathcal{T}_i(p) = \mathcal{T}_i^j(p)$ for all j.

We now briefly review the proposed approach in [16]. As shown in Table I, three "types" of players (that is, $\mathcal{B}$: bad, $\mathcal{N}$: new, and $\mathcal{G}$: good) with six possible outcomes are defined, where α and β determine boundaries on the trust values for different sets of players. This approach then applies functions $\mu(x)$ and $\mu'(x)$ respectively to update reputation of each player P_i, as shown in Figure 1. Parameters η, θ, and κ 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 due to our assumption in Definition 2.

We should stress that this function is not just a function of a single round, but of the "history". In fact, it rewards more the better a participant is, e.g., Figure 1: Cooperation, where $\mathcal{T}_i(p) \in [\alpha, 1-\epsilon]$, and penalizes more the worse a participant is, e.g., Figure 1: Defection, where $\mathcal{T}_i(p) \in [\epsilon-1, \beta]$. In addition, it provides opportunities for newcomers where we do not know much about their behaviors, e.g., Figure 1: where $\mathcal{T}_i(p) \in [\beta, \alpha]$.

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 I
SIX POSSIBLE ACTIONS FOR THE TRUST MANAGEMENT

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

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

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

Each function $\mu(x)$ and $\mu'(x)$ consists of four linear equations, each of which is simply determined by points (x_1, y_1) and (x_2, y_2) as follows: $y = \big((y_2-y_1)/(x_2-x_1)\big)(x-x_1)+y_1$.

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

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

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 need to satisfy conditions $1-\epsilon+\kappa \leq 1$ and $\epsilon-1-\kappa \geq -1$, or equivalently $\kappa \leq \epsilon$. This condition is sufficient to ensure that $\mathcal{T}_i(p)$ never exceeds $+1$ or -1.

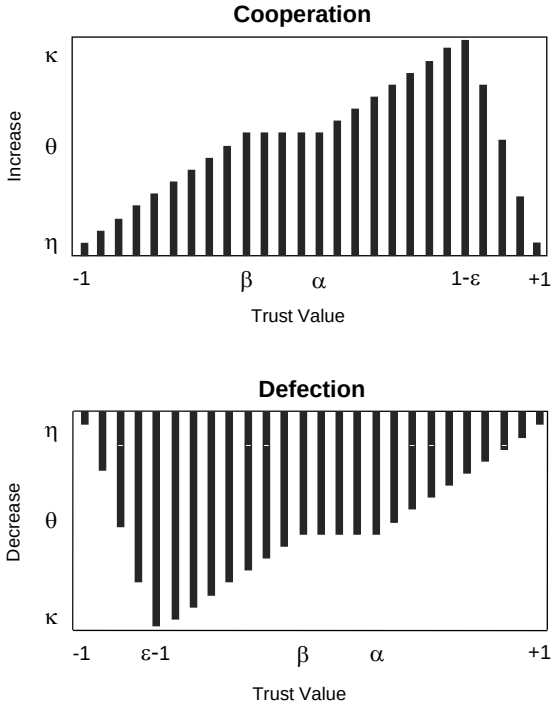

Fig. 1. Trust Adjustment by $\mu(x)$ and $\mu'(x)$ Functions

As shown, the "reward" and "penalize" factors are much larger than "encouragement" and "discouragement" factors. For instance, by assigning $\eta = 0.01 < \theta = 0.05 < \kappa = 0.09$, various points can be defined and an appropriate trust function can be constructed via regression.

It is worth mentioning that the authors in [16] also determine a 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.

III. Application in Cloud Computing

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 on their own. They can obtain these services from external providers according to their demands by a pay-per-use model [18], i.e., obtaining more services in the case of growing demand and vise versa.

A significant challenge in cloud computing is "resource management" due to the 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 [19].

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.
2) For the consumers to receive their services with a high satisfaction rate.

As an example, we can refer to excessive 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 takes an automatic configuration strategy and relies less on busier components during certain periods. We illustrate how this can be accomplished by continuous 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 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 $\vec{r} = (r_1, r_2, \ldots, r_n)$ and $\vec{w} = (w_1, w_2, \ldots, w_n)$ be the vector of players' trust values and the vector of players' weights accordingly. The initial values in $\vec{r}$ are going to be zero (i.e., all service providers are treated as newcomers), whereas the initial values in $\vec{w}$ are chosen by the dealer based on a specific distribution. We first define the following actions where each player's action $A_i \in \{\mathcal{C}, \mathcal{D}, \mathcal{X}\}$:

1) $\mathcal{C}$: for *cooperative* players where P_i is available at the required time and he sends correct shares to other parties.
2) $\mathcal{D}$: for *uncooperative* players where P_i is not available at the required time or he responds with delay.
3) $\mathcal{X}$: for *corrupt* players where P_i has been compromised by an adversary and he may send incorrect shares.

Let assume a secret key ξ 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 [20].

We can therefore assume that secret key ξ is used by many auctioneers to start or accomplish several sealed-bid auctions overtime 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. He then leaves the scheme.

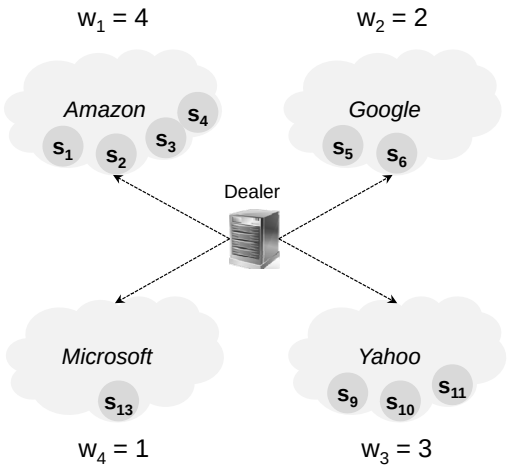

Fig. 2. System Initialization

Subsequently, different servers (or auctioneers) interact with the cloud providers to perform their tasks in the absence of the dealer, Figure 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.

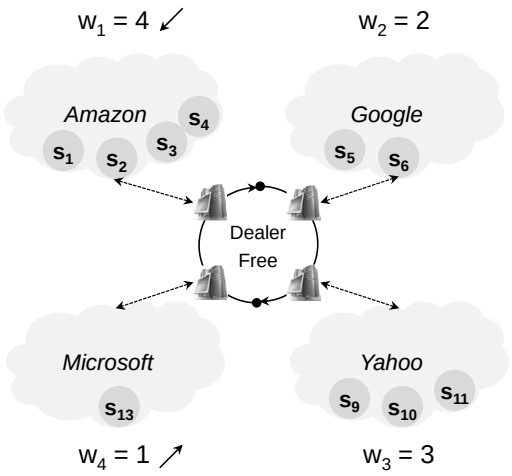

Fig. 3. Weight Adjustment

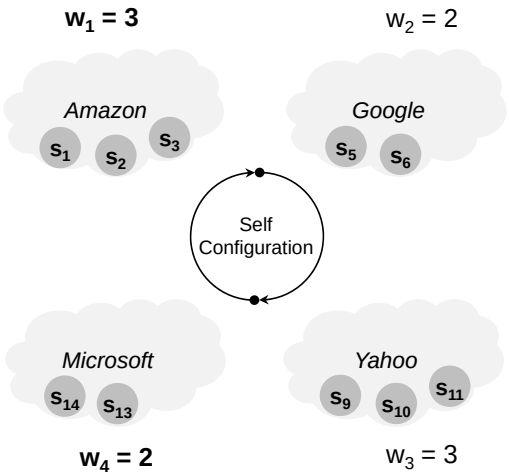

Fig. 4. Self-Configuration

Based on the service providers' actions $A_i \in \{\mathcal{C}, \mathcal{D}\}$ 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 3, the weights of two components are going to be updated. To see how amplification or reduction of a trust value affects the weight of a player, see [16] for *trust-to-share ratio* computation.

In the case of corruption $A_i = \mathcal{X}$, the corrupted providers are first rebooted. They then return to the scheme and are treated as newcomers. As we illustrated earlier, corrupted actions (e.g., sending incorrect shares) 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, shown in Figure 4. They initially enroll the new shares by using an *enrollment* protocol, e.g., suppose share s_{14} is enrolled for the fourth party. Subsequently, shares are updated (except the shares that are scheduled to be disenrolled) such that they are transformed to a new secret sharing polynomial, e.g., suppose share s_4 is not updated. Hence, the first player is going to have three shares afterward.

The benefit of using threshold secret sharing 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 participants can carry out the intented procedure. In the next section, we provide a new trust function that better fits to our model.

IV. New Trust Function

We would like to design a new trust function with social characteristics. The function that we reviewed in Section II uses the following two properties in order to adjust the trust value in different cases:

1) *Type*: parameters α and β 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 I.
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 α all the way to $+1$, which portrays how good a player is and what kind of history the player has.

Indeed, this trust function applies an *individual evaluation strategy*. We intend to use a *social evaluation strategy* by adding the following property to our new trust function:

3) *Sociality*: the new function takes into account the social behaviors of other parties each time it is used. 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. Let $\delta = \sum_{i=1}^{n} \ell_i$ denotes the total number of cooperative players. Intuitively, the new function should satisfy the following "social" conditions:

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

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

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

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

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

An example of the new social trust function is provided in Table II for further clarification. Each time the players "gain" partial of their rewards (e.g., 25%) that is proportional to the number of "non-cooperative" players. On the other hand, they "lose" partial of their trust value (e.g., 75%) that 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 II
COMPUTING $\mathcal{T}_i(p)$ WITH DIFFERENT VALUES OF δ

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 majority of the players are defecting and vise 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, the 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 actions and consequently competition elimination.

V. CONCLUSION

An application of social secret sharing in cloud computing was first demonstrated. Accordingly, a new trust function with social properties was proposed.

By providing a new model for system management in distributed secure schemes, 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 understanding of the existing models and their problems. As a result, this may lead to new solutions.

ACKNOWLEDGMENT

We would like to thank Urs Hengartner, Ian Goldberg, and other members of CrySP lab at the University of Waterloo for their helpful feedback on this research.

REFERENCES

[1] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[2] G. R. Blakley, "Safeguarding cryptographic keys," in *National Computer Conference NCC*. AFIPS Press, 1979, pp. 313–317.

[3] M. K. Franklin and M. K. Reiter, "The design and implementation of a secure auction service," *IEEE Transactions on Software Engineering*, vol. 22, no. 5, pp. 302–312, 1996.

[4] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable secret sharing and achieving simultaneity in the presence of faults," in *26th Annual IEEE Symposium on Foundations of Computer Science FOCS*, 1985, pp. 383–395.

[5] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *20th ACM Symposium on Theory of Computing STOC*, 1988, pp. 1–10.

[6] D. Chaum, C. Crépeau, and I. Damgård, "Multiparty unconditionally secure protocols," in *20th ACM Symposium on Theory of Computing STOC*, 1988, pp. 11–19.

[7] T. Rabin and M. Ben-Or, "Verifiable secret sharing and multiparty protocols with honest majority," in *21st Annual ACM Symposium on Theory of Computing STOC*, 1989, pp. 73–85.

[8] D. Beaver, "Multiparty protocols tolerating half faulty processors," in *9th Annual International Cryptology Conference CRYPTO*, ser. LNCS, vol. 435. Springer, 1989, pp. 560–572.

[9] D. R. Stinson and R. Wei, "Unconditionally secure proactive secret sharing scheme with combinatorial structures," in *6th Annual International Workshop on Selected Areas in Cryptography SAC*, ser. LNCS, vol. 1758. Springer, 1999, pp. 200–214.

[10] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin, "The round complexity of verifiable secret sharing and secure multicast," in *33th Annual ACM Symposium on Theory of Computing STOC*, 2001, pp. 580–589.

[11] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive secret sharing or: How to cope with perpetual leakage," in *15th Annual International Cryptology Conference CRYPTO*, ser. LNCS, vol. 963. Springer, 1995, pp. 339–352.

[12] R. Ostrovsky and M. Yung, "How to withstand mobile virus attacks: extended abstract," in *10th Annual ACM Symposium on Principles of Distributed Computing PODC*, 1991, pp. 51–59.

[13] J. C. Benaloh and J. Leichter, "Generalized secret sharing and monotone functions," in *8th Annual International Cryptology Conference CRYPTO*, ser. LNCS, vol. 403. Springer, 1988, pp. 27–35.

[14] M. Nojoumian, D. Stinson, and M. Grainger, "Unconditionally secure social secret sharing scheme," *IET Information Security, Special Issue on Multi-Agent and Distributed Information Security*, vol. 4, no. 4, pp. 202–211, 2010.

[15] M. Nojoumian and D. R. Stinson, "Brief announcement: secret sharing based on the social behaviors of players," in *29th ACM symposium on Principles of distributed computing PODC*, 2010, pp. 239–240.

[16] M. Nojoumian and T. Lethbridge, "A new approach for the trust calculation in social networks," in *E-business and Telecommunication Networks: 3rd International Conference on E-Business, Best Papers*, ser. CCIS, vol. 9. Springer, 2008, pp. 64–77.

[17] B. Yu and M. P. Singh, "A social mechanism of reputation management in electronic communities," in *4th Int. Workshop on Cooperative Info Agents CIA*, ser. LNCS, vol. 1860. Springer, 2000, pp. 154–165.

[18] W. Voorsluys, J. Broberg, and R. Buyya, *Cloud Computing: Principles and Paradigms*. John Wiley and Sons, 2011.

[19] J. Broberg, S. Venugopal, and R. Buyya, "Market-oriented grids and utility computing: The state-of-the-art and future directions," *Journal of Grid Computing*, vol. 6, no. 3, pp. 255–276, 2008.

[20] K. Peng, C. Boyd, E. Dawson, and K. Viswanathan, "Five sealed-bid auction models," in *the Australasian Information Security Workshop Conference, AISW'03*, vol. 21. Australian Computer Society, 2003, pp. 77–86.

VI. Appendix

A. Example of Threshold Secret Sharing

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

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

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

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

B. Example of Proactive Secret Sharing

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

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

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

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

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

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

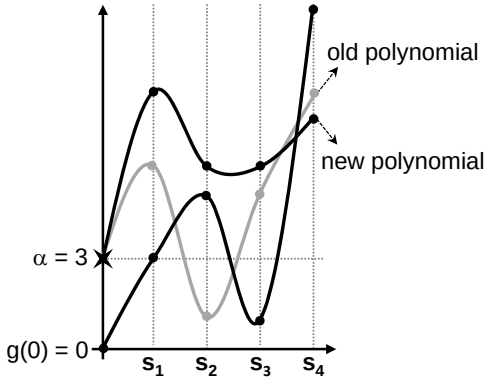

Fig. 5. Proactive Secret Sharing