Enabling Secure Secret Sharing in Distributed Online Social Networks

Le-Hung Vu, Karl Aberer
School of Computer and Communication Sciences,
École Polytechnique Fédérale de Lausanne (EPFL),
CH-1015 Lausanne, Switzerland
$\{$*lehung.vu* $|$ *karl.aberer*$\}$*@epfl.ch*

Sonja Buchegger
Deutsche Telekom Laboratories
TU Berlin, Germany
sonja@net.t-labs.tu-berlin.de

Anwitaman Datta
School of Computer Engineering
NTU Singapore
anwitaman@ntu.edu.sg

***Abstract*—We study a new application of threshold-based secret sharing in a distributed online social network (DOSN), where users need a means to back up and recover their private keys in a network of untrusted servers. Using a simple threshold-based secret sharing in such an environment is insufficiently secured since delegates keeping the secret shares may collude to steal the user's private keys.**

To mitigate this problem, we propose using different techniques to improve the system security: by selecting only the most reliable delegates for keeping these shares and further by encrypting the shares with passwords. We develop a mechanism to select the most reliable delegates based on an effective trust measure. Specifically, relationships among the secret owner, delegate candidates and their related friends are used to estimate the trustworthiness of a delegate. This trust measure minimizes the likelihood of the secret being stolen by an adversary and is shown to be effective against various collusive attacks. Extensive simulations show that the proposed trust-based delegate selection performs very well in highly vulnerable environments where the adversary controls many nodes with different distributions and even with spreading of infections in the network. In fact, the number of keys lost is very low under extremely pessimistic assumptions of the adversary model.

***Keywords*-secret sharing; online social networks; distributed online social networks; trust;**

I. Introduction

A threshold-based secret sharing scheme is a multi-party cryptographical protocol to enable a user to share her secret with only intended recipients in a distributed system [13]. A traditional (k, n)-threshold secret sharing protocol splits a secret into n parts (shares) any k of which (minimum) suffices to reconstruct the secret, e.g., the Shamir approach [14]. This approach can be well adapted to match the properties of peer-to-peer environment, and online social networking applications, where a user can not totally trust any other user, and no other single user is told the whole secret.

In this paper, we propose a new application of a threshold-based secret-sharing protocol in a *distributed online social network* (DOSN). Such a system uses a distributed or a P2P infrastructure for its users' data management and storage, while providing functionalities of conventional (centralized) social networking sites such as *Facebook.com* or *Orkut.com*. There are various motivations for such a decentralized architecture, foremost among these being users' privacy and autonomy from not only fellow users but also from service providers. The vision of DOSN platforms has been presented in several recent works, e.g., [1], [2], [5], [8], or *Tribler.org*.

We next describe our application of threshold-based cryptographical protocols through a concrete usage scenario. This scenario comes from our experience in the development of such a DOSN, where users need a means to back up and recover their private keys in a network of untrusted servers[1]. We also use this example to elaborate on the problem we study in this paper, as well as to define the scope of our intended solutions for the problem. A practical realization of this scenario and its related solutions is the recovery of user's passwords in a distributed storage system such as *Wuala.com*.

Private key recovery example: Alice's computer crashed, so she must use another computer. She wants to log in to her online social network (a DOSN) from the new computer, retrieve associated data and resume her life online.

When Alice first created her account using the previous computer, the system generated a private key as a means of authentication associated with her username. The private key of Alice is the ultimate secret enabling her to manage her personal data, e.g., to edit a blog entry or to configure her privacy setting. In contrast to conventional web-based online social networks such as *Facebook.com*, where user data are stored at servers owned by the service providers, a DOSN platform enables Alice to store her personal data *mainly on her computer* to ensure her total control on these data. However, anticipating a future crash of the original computer and loss of the data stored locally on it, and also to increase data availability, Alice's data is also encrypted and replicated in other machines. As the private key is difficult to remember and can also be lost, it is also backed up: Alice split the key according to a (2,3)-threshold cryptography approach, and stored that in the network itself.

For enhanced security, each part of the key was encrypted by a *passphrase* chosen by Alice, resulting in $n = 3$ encrypted shares. Each of the three delegates Bob, Carol,

This work was partly supported by the European Commission under the TEAM project (IST-35111-TEAM) and A-STAR grant No. 072 134 0055

[1] We will use two terms *secret* and *private key* interchangeably henceforth

and Dora is asked by Alice to keep a different encrypted share. These delegates are expected to only send a share to the user proven to be Alice.

Since the delegates may not know Alice a priori or they may not be able to meet in person, they need to verify Alice's identity (and thus ownership of the secret). Automatic verification, such as security questions/answers are applicable. These questions/answers can be possibly different for each delegate and digitally signed by Alice to prevent forgery. Upon successful verification of Alice's identity, a delegate sends back to Alice the locally stored encrypted share of Alice's secret. Alice recovers her private key by getting any two shares from three delegates, which she can decrypt using her private passphrase.

Unfortunately, there are several practical problems in using such a secret sharing scheme in a DOSN scenario.

- Users may be untrustworthy (*malicious*) when acting as delegates. An untrustworthy user may keep the shares to steal the secret for her own purposes, e.g., to control and steal Alice's private data. A user is untrustworthy either because she is curious or because her computer is controlled by a malicious software (an adversary). In the above example, the key of Alice is lost if (and only if) any $k \geq 2$ delegates among Bob, Carol, and Dora, are untrustworthy.
- The original secret can not be recovered without enough trustworthy delegates available, e.g., both Bob and Carol are on vacation and turn off their computers. Also, delegates may send invalid shares to reject the owner's requests, either intentionally or accidental due to software bugs, network errors, etc.
- The secret owner may forget the passphrase or answers to secret questions, and cannot recover the secret. The passphrase or these answers may also be lost (weak passwords) and thus an adversary can easily use this information to steal the secret (identity theft).

The main focus of this paper is on the first issue, as in online social networks the collusion among malicious delegates is even more feasible and detrimental. An adversary can control a large number of malicious users appearing as legitimate to coordinate the attack and steal the user's key, also performing a Sybil attack by assuming several identities. Having the private key, the adversary may tweak security options on the victim's machine, enabling malicious applications to control and use that machine for further attacks. The situation is even worse as a user usually trusts her friends, unaware of whether their machines are already under control of an adversary. This viral infection may spread through social links rapidly, potentially leading to an epidemic that at worst case makes the whole system eventually collapse.

Towards the above problem, most enhancements of threshold-based secret sharing schemes include the possibility to verify the validity of a share, to change the threshold dynamically, or to improve the computational and communicating efficiencies of the approach [13]. Current solutions to protect keys are to encrypt shares with passwords or to use verifiable credentials. The *resilience* of such protocols under *collusive attacks of malicious nodes* are not yet sufficiently studied. There has been *little study of how to select the most reliable delegates* under various adversary distributions in a large distributed network and understand the impact of such selection to the security level of the whole system. The main reason for such limitations is that *threshold cryptography is mostly used on systems under control of a single centralized provider*, which is different from our application context.

To improve security of the secret sharing protocol in such distributed scenarios, we propose in this paper a mechanism to select the most trustworthy delegates. Delegate trustworthiness is estimated by *exploiting relationships among the secret owner, delegate candidates, and their related friends*. This trust measure minimizes the likelihood of the secret being stolen by an adversary and is shown to be effective against various collusive attacks. Extensive simulation shows that compared to other approaches, e.g., [17], our trust-based selection performs very well in a variety of scenarios with several nodes under control of the adversary, with different distributions of adversarial nodes, and even with the spreading of infection from malicious nodes.

To the best of our knowledge, our approach is among the first ones applying threshold cryptographical protocols to enable secure secret sharing on distributed social networks. Our improved secret sharing scheme also has other practical applications, such as to enable delegated access control on other distributed systems. For instance, in a P2P-based content sharing system, a peer may rely on trustworthy delegates to distribute the data encryption key to those peers whose identities are unknown beforehand[2] yet proven to be from a subscribed reader group. With the given key, authorized readers can then decrypt any data replica by the original author even if the author is unavailable.

To reduce our work scope, we do not focus much on the second issue: the impacts of delegate availability to reconstruct the backed-up secrets. In fact, delegate unavailability is not a major problem in this key recovery scenario, as such recovery is assumedly unfrequent. Thus, if there are not enough delegates available at the moment to rebuild the secret, the owner may simply wait.

The third issue is related to the user's security awareness, which is orthogonal to our current problem. Nevertheless, with a threshold cryptographical approach, even if a user may choose weak passwords, an adversary must collect at least k shares and successfully decrypt them to steal the key. Therefore, using threshold cryptography for key backup is a generalized and more secured backup procedure compared

[2] Otherwise, a traditional PKI-based approach can be used, e.g., by encrypting the key with the public key of the authorized reader.

to conventional approaches, e.g., to backup the whole key on a single server on the network.

II. System model

A. Notations

Denote as $\mathcal{U}$ the set of users of an online social network. Let $\mathcal{D}_f$ be the set of possible types of relationships among users, e.g., family, close friends, colleagues, or acquaintances. Define the mapping $f : \mathcal{U} \times \mathcal{U} \to \mathcal{D}_f$ as relationships among these users $\mathcal{U}$. A distributed online social networking platform is formally defined as follows.

Definition 1: A *distributed online social network*, or *DOSN* for short, is a triple $\langle \mathcal{U}, f, \mathcal{RP} \rangle$, where $\mathcal{U}$ is the set of users or computers and f denotes the social relationships among them. The mapping $\mathcal{RP} : \mathcal{U} \to 2^{\mathcal{U}}$ is the data replication strategy that defines the set of nodes in the system to store data $\mathcal{R}_u$ of a user $u \in \mathcal{U}$.

Def. 1 associates each user with a node, e.g., her main working computer, in the underlying distributed storage system. This assumption simplifies our problem and subsequent analysis while still being realistic as most users primarily use one computer to work. Hence, we will use the notation peer, node, or user interchangeably in this paper. We are interested in building a mechanism to share a secret key in a DOSN platform, with any replication strategy, securely and effectively under various adversarial attacks.

B. Adversary model

We assume the adversary who wants to steal secrets of users to have the following capabilities.

A_1 The adversary can compromise many nodes (computers of users). Compromised nodes know and collaborate well with each other to achieve their goal: to steal as many secrets as possible.

A_2 Delegates of a user are publicly known and thus also known to the adversary.

A_3 The adversary has computational power to perform dictionary-attacks to decrypt the shares if she obtains it and can reconstruct the secret successfully. Furthermore, the adversary is cost-insensitive and has as much time as she wants to complete the attacks.

A_4 A user losing her secret may be infected and under control of the adversary. This leads to the spreading of infection (more peers become a bad choice as delegates) that may contaminate the whole network.

The assumption A_2 is to enable easy reconstruction of the secret without requiring each secret owner to remember her list of delegates. On the negative side, it reduces the attack cost of an adversary: she does not need to probe many users to steal a specific key. A practical system can put a few extra safe-guards such as not making the delegates public knowledge, at the cost of increased system design complexity. In summary, the above adversary is extremely powerful and our security analysis will be done under such pessimistic assumptions.

C. Security of the threshold-based secret sharing

The security and correctness of a (k, n)-threshold-based secret sharing protocol depend on the number of *trustworthy* and *corrupted* delegates in the delegate selection. A *trustworthy delegate* sends correct shares only to authorized requesters. Additionally, trustworthy delegates do not steal the secret by colluding with others to steal the original secret. A delegate who is not trustworthy is defined as *corrupted* or *malicious*.

Given the above adversary model, it is possible for an adversary to decrypt any encrypted share. Thus a (k, n)-secret-sharing scheme is secured if and only if there are: (1) at most $k - 1$ bad delegates; and (2) at least k trustworthy delegates available in a user's delegate selection to restore the secret. Therefore, the remaining and most important concern of a user is to choose her delegates to prevent corrupted delegates from stealing her secret by colluding with others. An approach to this problem is proposed and analyzed in Section III.

Throughout the paper, we rely on a number of other assumptions on our environments:

- A message from the owner to an off-line delegate can be pending. When online again, the delegate pulls all these off-line messages and processes them accordingly.
- We assume the compromise of a node by an adversary does not jeopardize its availability, similar to [17]. This is realistic since if infected machines become unavailable, e.g., cannot boot up or cannot connect to the network, this can signal to the owner to scan and clean her computer from malicious software.

III. Selection of reliable delegates

Due to various privacy and security settings, it is generally impossible for a user to crawl the whole network and gather all important information to best select the delegates. For example, personal data of a user and her relationships with others in most cases are not publicly available. Therefore, a user can only use her local knowledge and available public information in the network in making the delegate selection. We formally describe such a selection approach as follows.

Consider a DOSN $\langle \mathcal{U}, f, \mathcal{RP} \rangle$ as in Def. 1. Denote as $\mathcal{F}_u = \mathcal{F}_u^1$ the set of direct friends of a user u. The set of k-degree friends of u, where $k > 1$ is recursively defined as: $\mathcal{F}_u^k = \{w \mid w \in \mathcal{F}_v, v \in \mathcal{F}_u^{k-1}\}$. The set of all indirect friends of u is $\mathcal{F}_u^\infty = \bigcup_{k=1}^{\infty} \mathcal{F}_u^k$.

Let $\mathcal{P}_u^\infty$ be public personal information of users in $\mathcal{F}_u^\infty \subseteq \mathcal{U}$ and denote as f_u^∞ the set of connections among them. A (personalized) algorithm for a user u to select her delegates is given in Def. 2.

Definition 2: A *delegate selection algorithm* of a user u is defined as an algorithm operating on her personalized view

$\langle \mathcal{F}_u^\infty, f_u^\infty, \mathcal{P}_u^\infty \rangle$ on the social network and outputs a list of delegates $\mathcal{D}_u \in 2^{\mathcal{F}_u^\infty}$.

Let $\mathcal{D}$ be a set of delegates selected by u for a (k, n)-secret-sharing scheme[3]. Denote as $\mathcal{D}_c$ and $\mathcal{D}_a$ be respectively the number of corrupted and available delegates in the set $\mathcal{D}$. Also, define $\mathcal{D}_{at}$ the number of trustworthy and available delegates in $\mathcal{D}$. Our selection approach relies on the following concept of ε-security, given in Def. 3.

Definition 3: (ε-security) The selection $\mathcal{D}$ is said to be ε-secured if and only if the probability that at least k trustworthy delegates are available in $\mathcal{D}$ is $Pr(\mathcal{D}_{at} \geq k) \geq 1-\varepsilon$, where $0 < \varepsilon < 1$.

We want to study the way u selects her set of delegates $\mathcal{D}$ to ensure the availability and secured access to her backed-up secret. For simplicity, fix the number k and the security parameter $0 \leq \varepsilon \leq 1$. Let $\mathcal{D} = \{i, 1 \leq i \leq n\}$. We want to select $\mathcal{D}$ based on k, ε such that the resulting (k, n)- secret sharing scheme is ε-secured (Def. 3).

Denote $T(n, k-1) = Pr(\mathcal{D}_c \leq k-1)$ and $A(n, n-2k+1) = Pr(\mathcal{D}_a \geq 2k-1)$= Pr (at most $n-2k+1$ are offline). Assume that the probabilities that a user i is trustworthy and available, are $0 \leq t_i \leq 1$ and $0 \leq a_i \leq 1$, respectively. One can verify that $T(i, 0) = \prod_{j=1}^{i} t_j$ and $T(i, i) = 1$. The following recurrence relations can be obtained using basic probability update rules:

$$T(i{+}1, l{+}1) = t_{i+1}T(i, l{+}1) + (1{-}t_{i+1})T(i, l), \ 1 \leq i \leq n \quad (1)$$

Similarly, for $1 \leq i \leq n$, $A(i, i) = 1$, $A(i, 0) = \prod_{j=1}^{i} a_j$, and:

$$A(i+1, l+1) = a_{i+1}A(i, l+1) + (1 - a_{i+1})A(i, l) \qquad (2)$$

Given our adversary model in Section II-B, availability and trustworthiness of nodes are assumed to be independent. Therefore, the probability of at least k trustworthy delegates available among n delegates is:

$$\begin{aligned} Pr(\mathcal{D}_{at} \geq k) &\geq Pr(\mathcal{D}_a \geq 2k-1, \mathcal{D}_c \leq k-1) \\ &= A(n, n-2k+1)T(n, k-1) \qquad (3) \end{aligned}$$

Proposition 1 gives us certain properties of the probabilities $T(n, k-1) = Pr(\mathcal{D}_c \leq k-1)$ and $A(n, n-2k+1) = Pr(\mathcal{D}_a \geq 2k-1)$. The proof can be found in the appendix.

Proposition 1: For any $n \geq k > 0$:

(i) Irrespective of the selection $\mathcal{D}$, $T(n, k-1)$ and $A(n, n-2k+1)$, where $n > 2k-1$, are increasing functions of k and decreasing functions of n.

(ii) $T(n, k-1)$ is maximized where $\mathcal{D}$ is n delegates with highest trustworthiness t_i among the candidates.

A. Measuring trustworthiness of delegate candidates

Let i be a possible delegate candidate for u. In practice, i may be a friend of u, or a third-party provider offering data storage services. The measurement of the trustworthiness t_i of a user i is non-trivial. In our scenario, the notion of trust between two users is beyond the social trust between people, since the computer of a highly reliable and trustworthy friend may still be compromised by an adversary without the friend's awareness. Therefore, a user needs a more appropriate measure to evaluate the trustworthiness of a user before selecting her as a delegate. More precisely, t_i *is the personal belief of* u *on whether* i *is likely to be controlled by an adversary*. Such a value t_i depends on the following influential factors:

- *whether the node* i *is a well-known trusted entity*. For example, nodes from third-party providers offering data storage services can be seen as less vulnerable as they are usually equipped with up-to-date security patches and latest virus definitions.
- *whether* i *has potential to collude and steal a secret,* since curious friends may collude to get illegitimate access to unauthorized data. Also, if a user's friend is compromised by an adversary, other friends of hers are also vulnerable to attacks by the same adversary. In practice, viruses are likely to spread from one friend to another since people generally trust files or links sent by their friends. To minimize the influence of such attacks, we should give less trust to those delegates i with more chances to collude with each other. Informally, we should select delegates from different sets of friends to reduce the possibility of a collusive attack and minimize the influence of such a collusion.
- *whether* i *is an attractive target for an adversary*: since an adversary can minimize her attack cost by compromising nodes holding more keys, i is more attractive to an adversary if she is a delegate of many users. Hence, less trust should be put on those candidates i currently keeping more shares.

Given the above observations, the following heuristics can be used to evaluate the trustworthiness of a user. The key idea is to explore social relationships among users to prevent the spreading of infection from malicious nodes already under adversary control, as well as reducing the chance of colluding among these malicious nodes. More concretely, we define:

$$t_i = \begin{cases} 1 & \text{if } i \in D^0 \\ 1 - \delta(b_i) - \sigma(l_i) & \text{if } i \in [N] = \mathcal{F}_u \setminus D^0 \end{cases} \qquad (4)$$

where $b_i = \mid \mathcal{F}_i^\infty \cap \mathcal{F}_u \mid$ and l_i is the number of shares held by i. The candidate delegates $[N] = \{i, 1 \leq i \leq N\}$ are from the set of friends of the user $\mathcal{F}_u$. In Equation (4), D^0 is a set of $m < k$ preferred/trusted delegates chosen by the secret owner u. For instance, a good choice of D^0 include nodes from third-party providers offering data storage services.

Note that $b_i = \mid \mathcal{F}_i^\infty \cap \mathcal{F}_u \mid$ is the number of all indirect friends of i who are also in $\mathcal{F}_u \setminus D^0$. Thus $0 \leq \delta(b_i) \leq 1$ quantifies two influential factors: (1) how i is able to collude with her friends to compromise a shared secret; and (2) the influence i and her friends may have on the owner u if i is compromised by an adversary. Thus u puts higher trust on

[3] the index u is omitted for presentation clarity

a friend i with less connection with others of her friends. Given this observation, in our delegate selection algorithm, we set $k > \max_{i \in [N]}\{b_i\}$ (Algorithm 1).

The term $0 \leq \sigma(l_i) \leq 1$ determines the attractiveness of i to an adversary, e.g., how many keys the adversary gets by compromising i.

We let $0 \leq \delta(b_i) \leq 1$ reach the maximal value of 1 at $b_i \geq b^* = k - m$ and be at the minimum 0 at $b_i = 0$. Similarly, $\sigma(l_i)$ is defined to achieve the maximal value of $1 - \delta(b_i)$ at $l_i = l^* = \max\{l_i, i \in [N]\}$ and be 0 at $l_i = 0$. That means currently we care more on the impact of $\delta(b_i)$ than of $\sigma(l_i)$. The testing of different impact weights by $\delta(b_i)$ and $\sigma(l_i)$ is subject to future work. Depending on user's prior belief on the environment vulnerability, the following functions can be used:

- *exponential*: e.g., $\delta(b_i) = 1_{\{b_i > b^*\}} + 1_{\{b_i \leq b^*\}} \frac{2^{b_i}-1}{2^{b^*}-1}$ and $\sigma(l_i) = (1 - \delta(b_i))\frac{2^{l_i}-1}{2^{l^*}-1}$. These functions[4] are appropriate for *vulnerable* environments with potentially a lot of malicious users. Hence, the possibility that a node j would be compromised increases exponentially with the number of shares she keeps l_i and the number of common friends b_i between i and u.
- *logarithmic*: e.g., $\delta(b_i) = 1_{\{b_i > b^*\}} + 1_{\{b_i \leq b^*\}} \frac{log(b_i+1)}{log(m+1)}$ and $\sigma(l_i) = (1 - \delta(b_i))\frac{log(l_i+1)}{log(l^*+1)}$. These functions are appropriate in *more secured* environments with fewer malicious users, thus the possibility j being compromised increases less than linearly with l_i and b_i.
- *linear*: $\delta(b_i) = 1_{\{b_i > b^*\}} + 1_{\{b_i \leq b^*\}} \frac{b_i}{b^*}$ and $\sigma(l_i) = (1 - \delta(b_i))\frac{l_i}{l^*}$, which are applicable in *neutral* environments with a moderate number of malicious users.

B. A trust-based delegate selection algorithm

According to Proposition 1, the probability $T(n, k-1) = Pr(\mathcal{D}_c \leq k-1)$ is maximized by choosing $\mathcal{D}$ as n delegates with highest trustworthiness t_i from the candidates $[N]$. We assume that messages to unavailable delegates can be kept pending and processed later on when they go online (Section II). Hence the unavailability of delegates do not affect the computation of the probability $T(n, k-1)A(n, n-2k+1)$, and we can approximate $T(n, k-1)A(n, n-2k+1) = T(n, k-1)$.

With the trust measure in Section III-A, the selection of delegates to maximize the probability $T(n, k-1)$ is given in Algorithm 1. Roughly speaking, we sequentially pick a user i with the highest trust value t_i from remaining candidate delegates. Thus Algorithm 1 forms a delegate set $\mathcal{D}$ approximately maximizing $T(n, k-1)$. Since $Pr(\mathcal{D}_{at} \geq k) \geq T(n, k-1)$, we expect this algorithm to maximize the probability that the delegate set achieved the desired security level ε.

Algorithm 1 stops in two cases: first, all available N candidates are selected to be delegates; second, we achieve the desired security level $1 - \varepsilon$. In the first case, the actual security level of the selection is $Pr(\mathcal{D}_{at} \geq k) \geq Pr(\mathcal{D}_c \leq k-1) = T(n, k-1)$. This lower bound $T(n, k-1)$ also reflects the vulnerability of the environment and may be used by the user to decide whether to share her secret. In our later experiments, a user backs up her secrets only if the achieved security level is $Pr(\mathcal{D}_{at} \geq k) \geq \tau$, where $0 \leq \tau \leq 1 - \varepsilon$ is a parameter of our experiments.

Algorithm 1 *selectDelegates*(candidates $[N]$, trustworthiness t_i for each $i \in [N]$, threshold k, security parameter ε): selected delegates $\mathcal{D}$, achieved security level $T[n, k-1]$

```
$n = 0$; $\mathcal{D} = \varnothing$; $\mathcal{L} = [N]$; /* $\mathcal{L}$ is the current candidate list */
$T[n, 0] = T[n, 1] = 1$;
while ($n \leq N$ and $T[n, k-1] < 1 - \varepsilon$) do
    Pick user $i$ with highest $t_i$ in $\mathcal{L}$;
    $n = n + 1$; $\mathcal{D} = \mathcal{D} \cup \{i\}$; $\mathcal{L} = \mathcal{L} \backslash \{i\}$;
    $T[n, 0] = t_i T[n-1, 0]$; $T[n, k] = 1$;
    for $l = 1$ to $\min(n-1, k-1)$ do
        $T[n, l] = t_i T[n-1, l] + (1 - t_i) T[n-1, l-1]$;
    end for
end while
Return $\mathcal{D}$ and expected lower bound of security level $T[n, k-1]$;
```

IV. Experiments

The DOSN is simulated as a discrete event-based system with the agent-based simulation toolkit Repast [9]. A social network in an experiment is a graph of of 1028 users. The network topology follows Watts-Strogatz small world model [16], with connection radius 2 and rewiring probability 0.8 (thus this network has short average path length and a small clustering coefficient). This model was due to its simplicity and small world properties of social networks. Larger scale simulations are possible, yet we used smaller networks to reduce the simulation time. Another reason is that a DOSN is likely to be self-organized in ad hoc manner, e.g., among people within a geographical vicinity. Hence its size should be much less than traditionally centralized social networks. Experiments with other types of topologies are subject to future work (this is in fact a limitation of our work). We implemented a message-passing system where messages arriving at an unavailable recipient are made pending and processed later when the recipient goes back online. Such properties can be realized in practice with distributed storage techniques: messages are encrypted and stored in a DHT look up system [15] built on top of the user's computers. User unavailability hence, does not affect their being selected as delegates: a user sending a request for a share to an off-line delegate simply waits till the latter is available.

A. Implementation of delegate selection approaches

We implemented different delegate-selection approaches, each of which uses only local knowledge to select delegates, as in Def. 2.

[4] The function $1_{\{A\}}$ evaluates to 1 if A is true and to 0 otherwise.

- FRIENDBASED: a user selects all her friends as delegates. This approach was studied in a previous related work [17] for a different application (document signing) and in a limited case (without spreading of infection from malicious users).
- RANDOMWALK: a user random walks on the (public) social network graph and picks a delegate after $TTL = 7$ steps. This approach selects those delegates connected with many friends who also have high connectivity, e.g., nodes with higher PageRank-like values.
- TRUSTBASED: the trust-based delegation selection algorithm described in Algorithm 1, Section III. We used a reasonably small security parameter $\varepsilon = 0.001$, and with $\sigma(.)$ and $\delta(.)$ as exponential functions, i.e, users believe the environment is vulnerable. A user only decides to share a secret if and only if the achieved security level is $Pr(\mathcal{D}_{at} \geq k) \geq \tau$, where $0 \leq \tau \leq 1 - \varepsilon$ is an experiment parameter.

B. Experiment design

We use two following system performance metrics: f_L and f_b. $0 \leq f_L \leq 1$ is the fraction of secrets (private keys) lost among those backed up by using the threshold-based secret sharing scheme. A smaller f_L implies a more secured system and thus is preferable. $0 \leq f_b \leq 1$ is the fraction of users (among all) who can back up their secrets successfully. A higher f_b means higher usability of the system and thus is better. Note that f_b is only meaningful to the TRUSTBASED selection algorithm, since only this algorithm requires that a secret is backed up only if the achieved security level is $Pr(\mathcal{D}_{at} \geq k) \geq \tau$, where $0 \leq \tau \leq 1 - \varepsilon$. Other selection approaches (FRIENBASED or RANDOMWALK) have $f_b = 1$. We tested the TRUSTBASED approach with different values of τ and found that in most cases $f_b > 0.9$, and τ does not clearly affect f_L. Therefore, we focus on the performance metric f_L in later experiments with a given value $\tau = 0.75$.

Given the above metrics, the system load model consists of the following factors:

- inf: whether there is an infection spreading in the network, i.e., whether a user having lost her secret also becomes under adversary control.
- $pmal$: the percentage of initially malicious users, i.e., the number of users already under adversary control at the beginning of the simulation before the adversary commands them to initiate the coordinated attack. A higher $pmal$ means a more vulnerable environment.
- adv: the distribution of the initially malicious users in the network. The distribution can be random or based on certain criteria, e.g., number of friends of a user.

Beside the load factors and their intensities, the major factors influencing the system performance are:

- $select$: the algorithm to select delegates for the secret sharing protocol, which includes the three algorithms FRIENDBASED, RANDOMWALK, and TRUSTBASED as described in Section IV-A.
- $0 \leq \xi \leq 1$: the threshold k/n of a (k,n)-threshold secret-sharing scheme being used.

Our goal is to measure the effects of two factors $select$ and ξ to the system performance f_L under various load intensities inf, $pmal$, and adv.

We ran each simulation with appropriate parameters and measured each performance metric when the simulation reached the stationary regime. The result (not shown) confirms that the distribution of f_L is approximately Gaussian and perfectly iid. Therefore, for later experiments, we only ran each simulation with N=35 replications (sufficiently large sample size) and summarized the measurements of f_L with its means and confidence intervals at level 95%. As data is roughly normal iid, this summarization shows both accuracy and variability of our results.

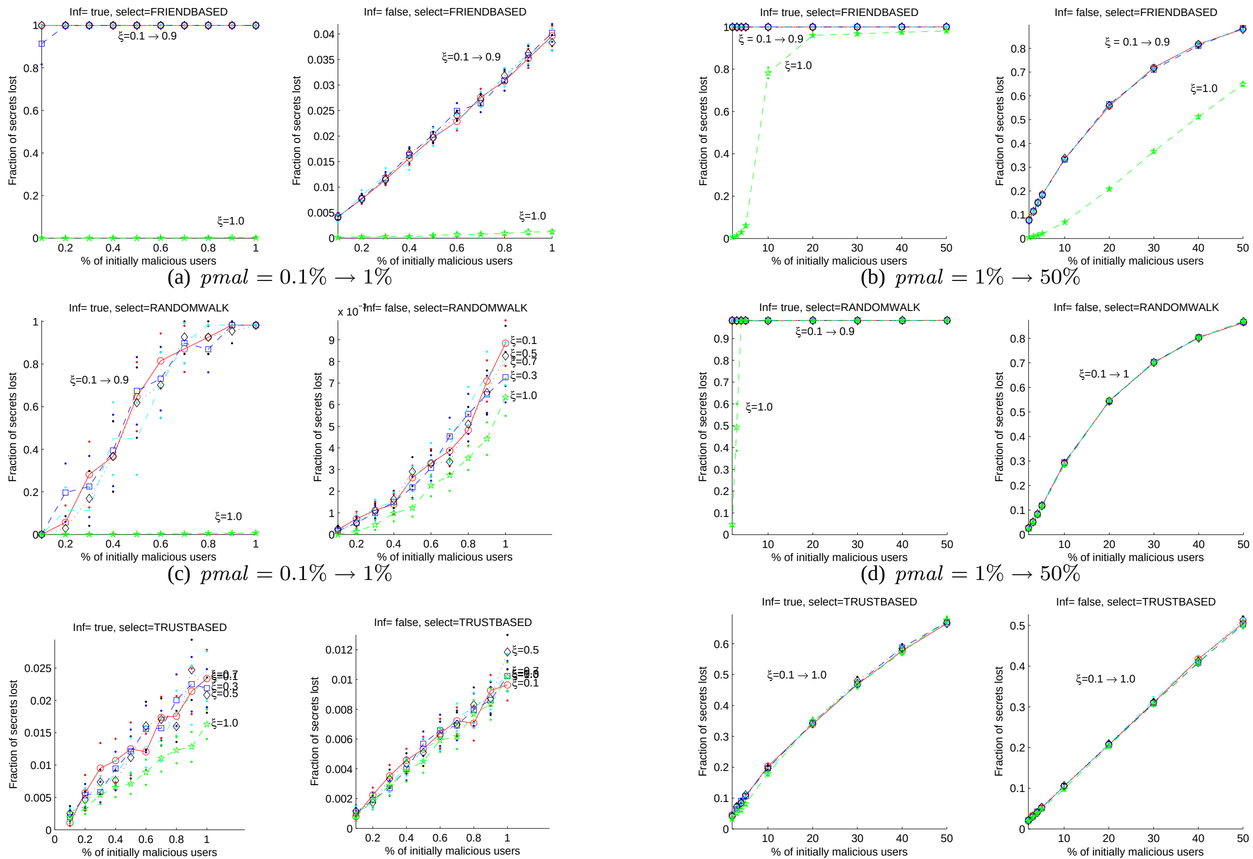
C. Effects of the threshold values ξ

We first measured influences of the cryptographical threshold $\xi = k/n$ to system security under various load intensities. For this goal ξ was varied from 0.1 to 1.0 for each of the delegate selection approaches FRIENDBASED, RANDOMWALK, and TRUSTBASED. The influence of ξ on the fraction of secrets lost f_L for each selection algorithm is given in Figure 1, where the Y-axis shows the mean of f_L and 95%-confidence intervals of the mean of f_L.

For each given delegation-selection algorithm, we varied the load intensity as follows. First, $pmal$ was increased from 0.1% to 50% to simulate environments with different numbers of malicious users under adversary control. To reduce the simulation parameter space, we randomized the distribution of these initially malicious users in the network (adv=RANDOM). Each experiment was carried out with both cases ($inf = false$ and $inf = true$).

We expected f_L to be lower with higher values of ξ. In fact, the results in Figure 1 give us two main observations. Firstly, for any delegate-selection algorithm and for any case of infection spreading, the smallest fraction of keys lost f_L is achieved with $\xi = 1$. This is intuitive: given a secret sharing scheme with $\xi = 1$, the adversary must successfully attack all delegates of a user to be able to steal the victim's secret. For values of $\xi < 1.0$ the differences are not substantial.

Secondly, we observe that the system security with an infection spreading (left pane of each figure) is much lower than without infection spreading (right pane). Figures 1(a, c, e), left panes, show that the adversary can steal a large fraction of secrets (from 50% up to 100% of secrets in most cases) by initially controlling a small number of malicious users (from 0.2% if $\xi < 1$ and from 5% if $\xi = 1$). The influence of the infection spreading is minimal in two cases: (1) with $\xi = 1.0$ and the delegate-selection approach FRIENDBASED as in Figure 1(a, left), and (2): for the TRUSTBASED delegation selection algorithm, as in



(a) $pmal = 0.1\% \rightarrow 1\%$

(b) $pmal = 1\% \rightarrow 50\%$

(c) $pmal = 0.1\% \rightarrow 1\%$

(d) $pmal = 1\% \rightarrow 50\%$

(e) $pmal = 0.1\% \rightarrow 1\%$

(f) $pmal = 1\% \rightarrow 50\%$

Figure 1: Influence of the crypto threshold ξ on the fraction of secrets lost f_L with different delegate-selection algorithms. The two critical values of ξ are $\xi = 1.0$ and $\xi < 1.0$

Figure 1(left side of e, f). These observations will be verified in the next section.

D. Effects of delegate selection algorithms

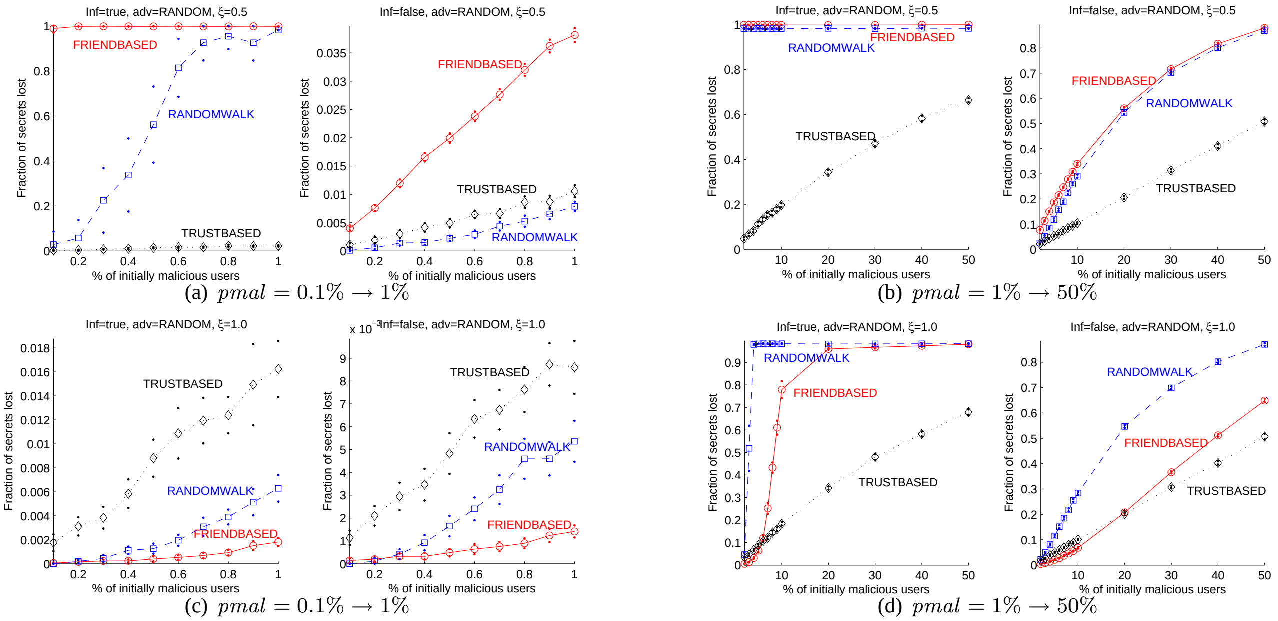
As the next step, we measured effects of different delegate-selection algorithms (FRIENDBASED, RANDOMWALK, and TRUSTBASED) to the system security under various load intensities.

Specifically, the first load factor $pmal$ was varied from 0.1% to 50% to simulate different environments with small (0.1% to 1%) and large (1% to 50%) numbers of malicious users under adversary control. Initially malicious users were randomly distributed in the network and each experiment was carried out in both settings: with or without infection spreading ($inf = false$ or $true$).

From previous experiments (Section IV-C), we are only interested in two critical values ($\xi = 1$ and $\xi < 1$) of threshold ξ. The results are given in Figures 2.

We observe that the selection approach TRUSTBASED is the best or comparable with the best in most cases, as shown in Figure 2(a, b, d). In the only case with a very small number of initially malicious users, and with $\xi = 1$ of Fig. 2(c), the selection algorithm FRIENDBASED is the best. In practice, using the maximal threshold $\xi = 1$ may not be preferable, as the secret owner may have to wait for every delegate to be online to reconstruct her backed-up secret. One possible reason why the TRUSTBASED selection algorithm performs less well in this case is the difference between the actual vulnerability of the environment and the use of exponential functions $\sigma(.)$ and $\delta(.)$ to evaluate the trustworthiness measure of available delegate candidates. Nevertheless, performance of the TRUSTBASED algorithm is still reasonably good in this case: $f_L < 0.018$ for $pmal \approx 1\%$ and $inf = true$ as in Fig. 2(c). Remember that f_L is our system performance metric measured under a very pessimistic estimation, where the adversary had unlimited computational power to decrypt any protected share it ever stole, the attained performance is acceptable.

The RANDOMWALK delegate selection approach, though intuitively appealing, is not superior. Its performance is somehow better where there is no infection spreading and the adversary controls a very small number nodes, as shown in Figure 2(c) and in the right pane of Figure 2(a).



(a) $pmal = 0.1\% \longrightarrow 1\%$

(b) $pmal = 1\% \longrightarrow 50\%$

(c) $pmal = 0.1\% \longrightarrow 1\%$

(d) $pmal = 1\% \longrightarrow 50\%$

Figure 2: Influence of delegate selection algorithms on f_L, distribution of $pmal$=RANDOM, $\xi = 0.5$ (a,b) and $\xi = 1$ (c,d)

Considering different adversary distributions: Another interesting question is how well different delegate selection algorithms perform under different adversary distributions. Specifically, we want to know whether the adversary may exploit her knowledge of the current delegate selection of users to focus her attack in a number of well-chosen users in the system.

Since the delegate selection algorithms FRIENDBASED, RANDOMWALK, and TRUSTBASED all use connectivity of users as a selection criteria for a delegate, an adversary may attempt the following attacks:

- LINKBASED: initially, the adversary tries to control nodes with higher numbers of links. These nodes are likely to be chosen by many users as their delegates, thus controlling these nodes help the adversary to steal more secrets at a lower cost.
- KEYBASED: the adversary focuses on compromising nodes currently keeping higher numbers of shares. This attack is even more powerful, as apparently controlling these nodes gives the adversary the highest likelihood of stealing the maximal number of secrets.

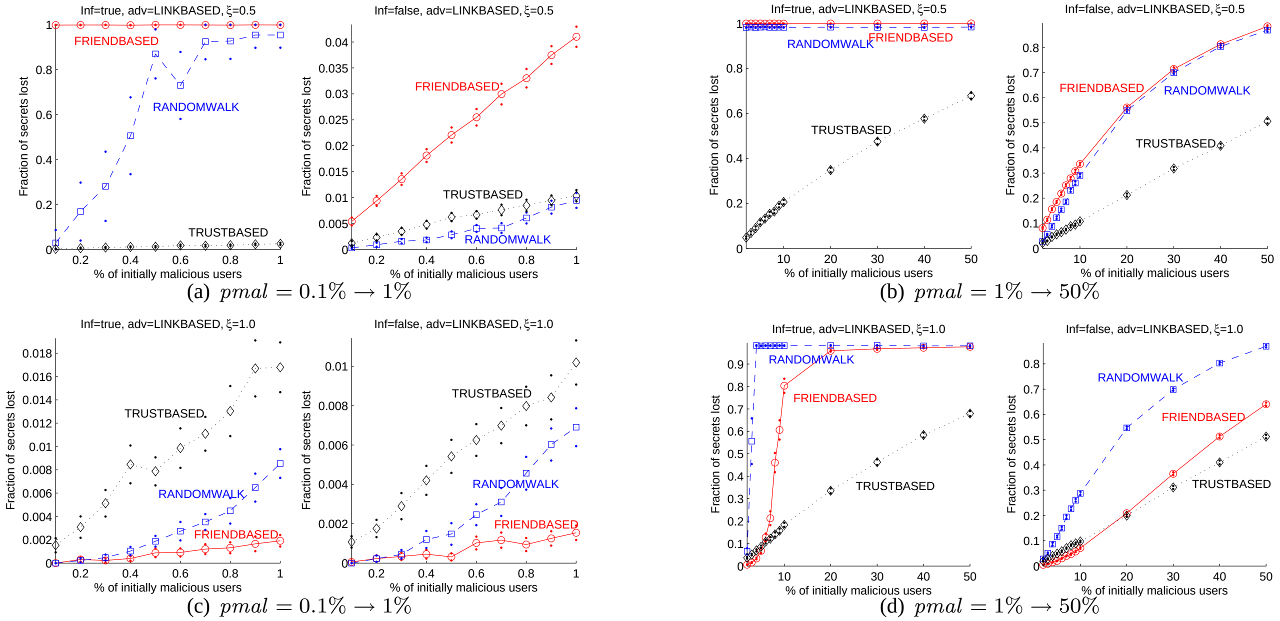
We performed various experiments to measure the influence of the above adversarial attack strategies. Load intensities are set up as in previous experiments, with varied numbers of initially malicious users: $pmal$ varied from small (0.1% to 1%) to larger (1% to 50%). Both cases of ($inf = false$ or $true$) are considered, each with two representative threshold values: $\xi = 0.5$ and $\xi = 1$.

Figure 3 shows the performance of different delegate-selection approaches under the adversary attack using the LINKBASED distribution. The result is similar to the previous case. The selection approach TRUSTBASED still performs either best or comparable to the best approach in almost all cases. Similar to the previous experiment with randomly distributed adversary, the TRUSTBASED algorithm performs less well than the FRIENDBASED approach where there is a very small number of initially malicious users and the threshold is $\xi = 1$. The algorithm RANDOMWALK is not superior in majority of cases we studied. Experiments for the adversary distribution KEYBASED are more complicated, as they requires detailed time-variant modeling of distributions of adversarial attacks and the selection of delegates. This experiment is subject to our future work.

V. Related work

Applications and usability of threshold cryptography in P2P and mobile ad hoc networks are discussed in detail by Saxena et al in [12]. This work mostly focuses on evaluating computational and communications overhead of different threshold cryptographical protocols. Security properties of such protocols under collusive attacks of many malicious nodes are not discussed.

There are several improvements of the original threshold-based secret sharing scheme, e.g., produce verifiable shares, change the threshold dynamically, improve the protocol's computational and communication efficiencies [13]. The selection of reliable delegates to minimize the influence of collusive attacks, however, has not been the focus of the security research community. Various security solutions mostly protect secret shares by encrypting these shares with passwords. These works are complementary to ours, since we can use any enhanced threshold cryptographical approach



(a) $pmal = 0.1\% \rightarrow 1\%$

(b) $pmal = 1\% \rightarrow 50\%$

(c) $pmal = 0.1\% \rightarrow 1\%$

(d) $pmal = 1\% \rightarrow 50\%$

Figure 3: Influence of delegate-selection algorithms on f_L, distribution of $pmal$=LINKBASED, $\xi = 0.5$ (a,b) and $\xi = 1$ (c,d)

for our delegated key recovery scenario.

To the best of our knowledge, our approach is the first one applying threshold cryptography to enable private key backup and recovery in distributed social networks. Another novelty of this work is the proposal of appropriate trust measures based on social relationships among users. This solution is shown to be effective against possible infection spreading, and resilient under presence of various malicious nodes with different distributions.

The work most related to ours is [7], [17], where the authors apply threshold cryptographical approaches for a different problem on social networks. [17] considers the application of threshold cryptographical protocols for signing documents, which is different from ours since it requires a user to be available to issue the digital signature. Regarding the technical approach, our work is also different from [7], [17]. First, we study the effectiveness of many delegate selection approaches given the presence of various numbers of malicious users and with possible infection spreading. The FRIENDBASED delegate selection algorithm that we analyzed is the same as the approach of using trusted friends to store the cryptographical keys proposed by [17]. Second, the adversary model we are considering is more powerful with its capability of spreading its infection to many nodes. Our simulations also consider many possible attack models of the adversary. [7] proposes to create certificates based on agreement of a number t of nodes in the network determined dynamically. The main result is a preliminary analysis on the probability of nodes being attacked. The issues of infection spreading and the selection of reliable delegates, however, are not yet studied.

In a wider context, our work is also related to credential-based access control in P2P and social networks. Credentials can be generated by specialized hardware such as the Trusted Computing Platform [11], or based on certain attributes of the requesters [10]. Most work in this direction addresses the general problem of access control with the assumption that access is granted to authorized users with identities known beforehand. This assumption is not valid in our scenario of backing up and recovery of secrets and thus these approaches are not applicable. In other works, such as [3], [4], [6], [10], heuristic measures are used to evaluate trust between the resource owner and the requester before the credential is granted. For example, [3] assigns each edge between users a trust level and a relationship type and uses a central node for registration and management. [6] proposes to grant access to resources based on chain of trust relationships between the resource owners and the requesters. An open problem of these solution approaches is a detailed analysis of their correctness and security under collusive attacks and spreading of adversarial infection in the network.

VI. Conclusion and Future Work

The question of secure backup and recovery of secrets in (distributed online social) systems is an important and challenging issue. Our work provides a promising first step toward an appropriate technical solution to this problem. We have studied the application of threshold-based secret sharing protocols for this purpose and suggested potential mechanisms to improve the security of these protocols. Specifically, we have proposed an algorithm to select the most reliable delegates to enable such a secure secret sharing in a network of untrusted nodes. This approach selects delegates based on a simple trust measure that exploits social relationships among the secret owner, delegate candidates, and their friends, to minimize the probability that a set of delegates collude to steal the secret. Such an approach has been shown to perform very well under a variety of scenarios, even with a large number of initially malicious users with possible spreading of infection in the network.

A limitation of this work is that we only consider a small-world network topology with homogeneous degree distribution. It would be important to also measure and analyze performance of various selection algorithms in larger scale heterogeneous networks, e.g., with heavy-tailed node degree distributions. The dynamics of networks with a growing number of users is also an important factor yet to be studied.

As part of our future work, we plan to study realistic cost models of an adversary to integrate with the design of potential delegate selection mechanisms. This issue is important since the attack by a rational adversary may be well-related to its endured cost. The self-healing capability of the distributed social network via the detection and cleaning of malicious and infected nodes is also an interesting issue we plan to study as part of future work.

References

[1] S. Buchegger and A. Datta. A case for P2P infrastructure for social networks - opportunities and challenges. In *Proc. of the 6th International Conference on Wireless On-demand Network Systems and Services*, Feb, 2009.

[2] S. Buchegger, D. Schiöberg, L. H. Vu, and A. Datta. PeerSoN: P2P social networking - early experiences and insights. In *Proc. of 2nd ACM Workshop on Social Network Systems Social Network Systems, 2009*.

[3] B. Carminati and E. Ferrari. Privacy-aware collaborative access control in web-based social networks. *DBSec, 2008*.

[4] S. Chakraborty and I. Ray. TrustBAC: integrating trust relationships into the rbac model for access control in open systems. In *Proc. of the 11th ACM symposium on Access control models and technologies (SACMAT'06*, 2006.

[5] L. A. Cutillo, R. Molva, and T. Strufe. Privacy preserving social networking through decentralization. In *Proc. of the 6th International Conference on Wireless On-demand Network Systems and Services*, Feb, 2009.

[6] J. Domingo-Ferrer, A. Viejo, F. Sebe, and U. Gonzalez-Nicolas. Privacy homomorphisms for social networks with private relationships. *Computer Networks (in press, available online 11 July 2008)*, 2008.

[7] c. L. Fran L. Mé, and V. V. T. Tong. A distributed certification system for structured P2P networks. In *Proc. of the 2nd International Conference on Autonomous Infrastructure, Management and Security*, 2008.

[8] C. A. Yeung, I. Liccardi, K. Lu, O. Seneviratne, and T. Berners-Lee. Decentralization: The future of online social networking. In *W3C Workshop on the Future of Social Networking Position Papers*, 2009.

[9] M. J. North, N. T. Collier, and J. R. Vos. Experiences creating three implementations of the repast agent modeling toolkit. *ACM Trans. Model. Comput. Simul.*, 16(1):1–25, 2006.

[10] E. Palomar, J. M. E. Tapiador, J. C. Hernandez-Castro, and A. Ribagorda. Secure content access and replication in pure p2p networks. *Comput. Commun.*, 31(2):266–279, 2008.

[11] R. Sandhu and X. Zhang. Peer-to-peer access control architecture using trusted computing technology. In *Proc. of the 10th ACM symposium on Access control models and technologies (SACMAT'05)*, 2005.

[12] N. Saxena, G. Tsudik, and J. H. Yi. Threshold cryptography in p2p and manets: The case of access control. *Comput. Netw.*, 51(12):3632–3649, 2007.

[13] B. Schneier. *Applied cryptography (2nd ed.)*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[14] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[15] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01*, pages 149–160, New York, NY, USA, 2001. ACM Press.

[16] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

[17] S. Xu, X. Li, and P. Parker. Exploiting social networks for threshold signing: attack-resilience vs. availability. In *ASIACCS '08: Proc. of the 2008 ACM symposium on Information, computer and communications security*, 2008.

Proof of Proposition 1

(i) We prove the monotonicity of $T(n, k-1)$ first. The proof for $A(n, n-2k+1)$ can be done similarly. From its definition, it follows that $T(n,k) \geq T(n,k-1)$. The proof of $T(n+1,k) \leq T(n,k)$ is also trivial. In fact, $T(n+1,k) - T(n,k) = t_{n+1}T(n,k) + (1-t_{n+1})T(n,k-1) - T(n,k) = (1-t_{n+1})(T(n,k-1) - T(n,k)) \leq 0$, since $T(n,k-1) \leq T(n,k)$ from the above result.

(ii) Suppose that the delegate set $\mathcal{D}$ has n delegates, where each $i \in \mathcal{D}$ has $t_i \geq t_j, j \in [N] \setminus \mathcal{D}$.

Consider another delegate set $\mathcal{D}'$ different from $\mathcal{D}$ by one delegate. Define $\mathcal{D}' = succ(\mathcal{D}) = (\mathcal{D} \setminus \{i\}) \cup \{j\}$, where $j \in [N] \setminus \mathcal{D}$ and $t_i \geq t_j, 1 \leq i \leq n-1$. We will show $Pr(\mathcal{D}_c \leq k-1) \geq Pr(\mathcal{D}'_c \leq k-1)$.

As delegates have equal roles, the order of putting a user i in a delegate set D does not influence the probability $Pr(\mathcal{D}_c \leq k-1)$. Let us select the user i at the last step to form the set $\mathcal{D}$ and select j at the last step to form $\mathcal{D}'$.

It follows that $Pr(\mathcal{D}_c \leq k-1) = t_i T(n-1,k-1) + (1-t_i)T(n-1,k-2)$ and $Pr(\mathcal{D}'_c \leq k-1) = t_j T(n-1,k-1) + (1-t_j)T(n-1,k-1)$. Since $T(n-1,k-1) \geq T(n-1,k-2)$ according to (i), and $t_i \geq t_j$ from the definition of $\mathcal{D}'$, we have:

$$\begin{aligned} Pr(\mathcal{D}_c \leq k-1) - Pr(\mathcal{D}'_c \leq k-1) &= (t_i - t_j)(T(n-1,k-1) \\ &\quad -T(n-1,k-2)) \\ &\geq 0 \end{aligned}$$

More generally, given any selection $\mathcal{D}''$ different from $\mathcal{D}$ by $J \leq n$ users, one can verify that there exist a series of delegate selection $\mathcal{D}^j, 1 \leq j \leq J$ such that $\mathcal{D}'' = \mathcal{D}^J, \mathcal{D}^{j+1} = succ(\mathcal{D}^j), 1 \leq j \leq J-1$, and $\mathcal{D}^1 = succ(\mathcal{D})$. Since $\mathcal{D}' = succ(\mathcal{D})$ implies $Pr(\mathcal{D}'_c \leq k-1) < Pr(\mathcal{D}_c \leq k-1)$, it follows that $Pr(\mathcal{D}''_c \leq k-1) < Pr(\mathcal{D}_c \leq k-1)$. Therefore, $Pr(\mathcal{D}_c \leq k-1) = T(n,k-1)$ is maximized with the delegate set $\mathcal{D}$.