DSO: Dependable Signing Overlay[1]

Guofei Gu, Prahlad Fogla, Wenke Lee, Douglas Blough
{guofei, prahlad, wenke}@cc.gatech.edu, doug.blough@ece.gatech.edu

Georgia Institute of Technology

Abstract. Dependable digital signing service requires both high fault-tolerance and high intrusion-tolerance. While providing high fault-tolerance, existing approaches do not satisfy the high intrusion-tolerance requirement in the face of availability, confidentiality and integrity attacks. In this paper, we propose Dependable Signing Overlay (DSO), a novel server architecture that can provide high intrusion-tolerance as well as high fault-tolerance. The key idea is: replicate the key shares and make the signing servers anonymous to clients (and thus also to the would-be attackers), in addition to using threshold signing. DSO utilizes structured P2P overlay routing techniques to provide timely services to legitimate clients. DSO is intended to be a scalable infrastructure for dependable digital signing service. This paper presents the architecture and protocols of DSO, and the analytical models for reliability and security analysis. We show that, compared with existing techniques, DSO has much better intrusion-tolerance under availability, confidentiality and integrity attacks.

Key words: intrusion-tolerance, fault-tolerance, P2P overlay, dependable, digital signing service.

1 Introduction

Digital signing is an integral part of a modern computer security architecture. It is one of the basic services provided by any CA (Certificate Authority) or PKI (Public Key Infrastructure) system. Dependable digital signing service should continue to provide service despite system failures and malicious attacks. In other words, it needs to be both high fault-tolerant and high intrusion-tolerant (also called attack-tolerant).

Researchers have been studying techniques to provide both fault-tolerant and intrusion-tolerant service (not limited to signing service) using multiple servers. Traditional fault-tolerant approaches (e.g., replication and Byzantine quorum systems [8, 18]) mainly provide redundancy. Secret sharing [25, 6, 9] and threshold cryptography [5, 21, 26] can be used to provide certain level of fault-tolerance and intrusion-tolerance. For instance, a (k, m) threshold scheme divides a secret into m pieces such that any k or more pieces can be used to reconstruct the secret. Knowledge of any $k - 1$ or fewer pieces does not provide any information of the secret. These threshold schemes can provide intrusion-tolerance up to a given threshold, above which there is no security at all. For example, using a (k, m) scheme can tolerate confidentiality and integrity attacks up to $k - 1$

[1] A slightly shorter version without dynamic intrusion-tolerance analysis appeared in ACNS'06.

shares. If k or more shares are compromised, the attacker can reconstruct or modify the secret data successfully. Similarly, if $m - k + 1$ or more nodes are under an availability attack (e.g., Denial of Service, or DoS), then the signing service becomes unavailable.

We propose a novel architecture, Dependable Signing Overlay (DSO), to enhance intrusion-tolerance and fault-tolerance for digital signing service. The key idea is: replicate the key shares and make the signing servers anonymous to the other hosts including the clients, in addition to using threshold signing. The threshold signing scheme and replication technique provide fault-tolerance. The threshold signing scheme and anonymous signing servers provide intrusion-tolerance because the attackers cannot know which signing servers to attack in order to deny signing service, steal, or corrupt the secret signing key. By systematically combining these three techniques, DSO can provide not only very high fault-tolerance but also very high intrusion-tolerance. Although DSO technique can be extended as a general architecture to provide other dependable services, we focus our effort on providing a scalable infrastructure or platform for dependable signing service in this paper. The design and evaluation of DSO are presented. Specifically, we make the following contributions:

– **Architecture and Protocol Design** A key goal in DSO is to make the signing servers anonymous, or "hidden" among a large number of DSO nodes, so that all an adversary can do is just to randomly attack some nodes on DSO. Thus, the chance of a successful attack on confidentiality, availability, and integrity is low. On the other hand, an important goal is that legitimate client requests are served correctly and in a timely manner. We accomplish these goals by designing DSO as a P2P (peer-to-peer) overlay server network and adopting the techniques of structured P2P overlay routing based on DHT (Distributed Hash Table). Section 3 presents the architecture and main protocols of DSO.
– **Reliability and Security Analysis** We derive analytical models so that we can concretely analyze and evaluate the reliability (fault-tolerance) and security (intrusion-tolerance) of DSO. The security analysis considers confidentiality, availability, and integrity attacks under both static and dynamic (with recovery) situations. Our results show that DSO provides very high fault-tolerance. For example, if the reliability of a single node is just 0.6 and we use a (6,10) threshold scheme in a 100 node DSO, the reliability of the service is more than 0.999. DSO also provides very high intrusion tolerance. For example, using a (6,10) threshold scheme in a 100 node DSO, when 30 nodes are attacked, the probability of successfully compromising availability is only 0.0000034754. The details of the analysis are in Section 4.
– **Comparison with Existing Schemes** We show that, compared with existing representative techniques, DSO provides high fault-tolerance and much higher intrusion-tolerance. For example, a (6,10) threshold signing scheme can tolerate only up to 5 compromised nodes. If we use a (6,10) threshold signing scheme in a DSO with 100 nodes and an attacker compromises 30 nodes, the probability that the attacker obtains the signing key is just 0.0473. Section 5 discusses the comparison of DSO with existing schemes.

2 Related Work

Fault-tolerance and intrusion-tolerance have been studied intensively by the research community. Traditionally, fault tolerance is addressed by replicating the service and building quorum systems [8, 18].

Different threshold schemes were designed to build intrusion-tolerant services which can tolerate successful intrusions on less than k servers. Shamir's secret sharing [25] is a simple threshold scheme based on polynomial interpolation. Later, verifiable secret sharing schemes were proposed in [6, 4] for the verification of secret shares by share holders and share combiner. Proactive secret sharing schemes [9, 15] were proposed for the renewal of the shares without reconstructing the secret. Unlike secret sharing schemes that require secret data to be reconstructed at a trusted host, Threshold cryptography [5, 21, 26, 10] can use or generate secret key in a distributed fashion without the need of any trusted host. They used to provide decryption service, signing service and other CA services. The Intrusion Tolerance via Threshold Cryptography (ITTC) project at Stanford [29] used threshold cryptography to provide basic public key services without ever reconstructing the key. Zhou *et al.* and Luo *et al.* have applied threshold cryptography into Ad-hoc networks [30, 14, 17] to provide CA service to solve key management and membership problems. Narasimha *et al.* [19, 20] used threshold cryptography in P2P and MANET to provide efficient member control and node admission.

Recently, [31, 16, 2] tried to use traditional fault-tolerance techniques (e.g. replications, Quorums) along with secret sharing or threshold cryptography to provide both high fault-tolerance and intrusion-tolerance. Cornell Online CA (COCA [31]), combined quorum and threshold cryptography techniques to provide a secure distributed certificate authority. Lakshmanan [16] proposed a scheme to combine replication with secret sharing to provide secure and reliable data storage. MAFTIA [2] is a comprehensive approach for tolerating both accidental faults and malicious attacks in large-scale distributed systems.

One main problem with the above described systems is that it is easy for an attacker to find the address of the servers used in the system and directly attack them. A possible solution is to use anonymity and randomness techniques, as seen in some censorship-resistant publishing systems like Eternity Service [1], Publius [28], free haven [7]. These systems focus on providing distributed document storage and censorship-resistant publishing, which is different from the target of DSO. Secure Overlay Service (SOS [12]) also used the anonymity and randomness of the overlay network to make it difficult for the attacker to target any particular node for DoS attacks. We also try to exploit the anonymity provided by structured peer-to-peer networks to hide the actual service providers from the attacker. To take over any service, an attacker needs to randomly attack the nodes in the overlay network and hope for the best. In DSO, we couple service provider anonymity with threshold schemes and replication to provide a highly secure digital signing service which has both high reliability and security.

3 Architecture and Protocol Design

3.1 Design Goals

DSO aims to provide dependable digital signing service that can tolerate a large number of faulty nodes or malicious attacks. This security goal is achieved by making the signing servers anonymous, so that an attacker does not know which signing servers to attack. This reduces the chance of successful attacks on the system. Even though the servers are anonymous, legitimate client requests should be served efficiently. We choose structured P2P network for the architecture because it provides anonymity of the servers with minimal network overhead. An initial signing server generates the pair of public and secret keys, then distributes the secret key (signing key) to a number of nodes in the overlay using secret sharing. Later, a client requesting the signing service may obtain the partial signature from required nodes and reconstruct the signature using threshold signing scheme.

3.2 Structured P2P

Recently, many services are deployed on peer-to-peer networks. P2P is a fault-tolerant network because it provides redundancy (through replication) and it can automatically adapt itself to the failure or arrival/departure of nodes. Current structured P2P systems (e.g. Chord [27], Pastry [23]) are based on distributed hash table (DHT) technique to efficiently route the packets from source to destination. A typical structured P2P system based on DHT provides the following guarantees:

- Communication to any peer (or query to any data) identified by some key ID (a hash value) is guaranteed to succeed if and only if the peer (or data) corresponding to the key ID is present in the system;
- Communication to any peer (given ID) is guaranteed to terminate within a small and finite number of hops;
- The key ID space is uniformly divided among all currently active peers;
- The system is capable of handling dynamic peer joins and leaves.

In DSO, we can use a routing technique similar to Chord [27], which guarantees that a packet will get to its destination in no more than $log(N)$ hops (N is the size of the overlay) by looking up the ID (a hash value) of the destination. We can create multiple destination nodes for a given identifier by using multiple hash functions. By carefully choosing the proper class of hash functions, the sequences of nodes used to route a packet from a node to the destination can be independent from one another. Chord is robust to changes in overlay membership: each node's list is adjusted to account for nodes leaving and joining the overlay. This is called the self-healing feature of structured P2P networks. Note the original Chord directly maps a node's IP address to the ID using a hash function. Given the size of network (typically smaller than a million), an attacker can calculate the ID of each host in the network and store it. Now given the ID, he can easily determine the address of the host. This is not safe in DSO because our security guarantees rely on the anonymity of the servers in the network: no one should be able to convert node's ID to its IP address. Thus we do not use the original Chord's node identifier mapping mechanism. We can use a Pastry-like mechanism [23] (node IDs are assigned randomly with uniform distribution from a circular ID space) for DSO that makes it very hard to determine the IP address of the host given its ID. At the same time we should take care of the security of node ID assignment as in [3].

3.3 System Architecture and Process

The basic system architecture is shown in Figure 1. The three main components are share holders (SH), beacons and access points (AP). Brief functionalities of these three components are described below. Details are in Section 3.4.

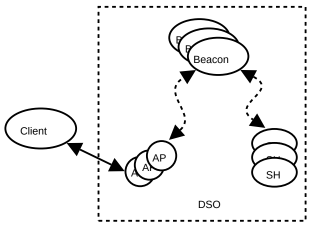


Fig. 1. Dependable Signing Overlay (DSO)

DSO can provide multiple signing services at the same time. Any signing service is recognized using a service key tag. Each service has its own secret signing key, also called service key. This private key is divided into m parts using a secret sharing scheme (k, m). Each of these m shares are replicated to n_h distinct hosts selected by the initial signing server. Thus, there are n_h copies of each share of the service key in the system. All these shares are sent to selected SHs by the initial signing server.

On receiving the share, a share holder calculates the index keys using a number of (n_b) well-known consistent hash functions as in SOS[12] operated on the service key tag. These index keys will identify the IDs of a set of overlay nodes that will act as beacons. Beacons will be contacted and store IDs of these SHs at the end of service initialization process (see Section 3.4).

Certain nodes in DSO act as Access Points (AP) which have the capability to authenticate clients. Before acquiring signing service, a client should first contact an overlay AP to request a certain signing service (indicated by the service key tag). After authenticating and authorizing the request, the AP securely routes the request from the client to the correct beacons (whose IDs are identified with hash values of the service key tag) using the underlying routing mechanisms. (Each node in the path determines the next hop by applying appropriate hash function to the service key tag.)

After a beacon receives the signing request, it will route the packet to the proper share holders which will perform threshold signing and produce partial signatures according to the request. These partial signatures are then combined by a beacon to produce the final signature. The combiner (beacon) then sends the result to AP. The result is finally forwarded to the client. During the whole process, the original secret (private key) is never disclosed to anyone. The message sent from the client can be blinded (see Section 3.4) to achieve confidentiality.

This scheme is robust because:

- Each secret in the system is shared by different nodes and all of the shares are replicated to multiple servers. This provides good redundancy while maintaining the secrecy of the data.
- There are multiple access points, beacons, and share holders on DSO for every service. Any node in this overlay can be AP, beacon or share holder. In fact, in our discussion we assume all nodes are access points.
- As there are lots of APs available, failure of one AP does not have much effect on the system. Client can simply choose another AP to enter the overlay.
- The communication to some target node given its ID (not IP address) is most likely not direct, but indirect through a set of intermediate peer forwarding. This provides anonymity.
- If some beacon fails, DSO service can self-heal by choosing a new node as the beacon with using some new hash function.
- If some share holders are compromised or targeted by attackers, the service provider can choose an alternate set of share holders and such operations are transparent to clients.
- All the beacons and SHs are anonymous to the clients (clients do not know the IP address of any SH or beacon for a service). Even if some beacon is compromised, the attacker can only get the IDs of SHs. As we mentioned above, with only IDs an attacker cannot know the real addresses. Thus, the SHs are still anonymous to attackers even though they know the IDs. Similarly, if any access point is compromised, the attacker can only get the IDs of beacons, but not the real address.

3.4 Protocol Design

In this paper, we choose threshold RSA signing [26, 19, 20, 10] as our basic signing scheme because it is one of the most popular schemes (one can also use threshod DSA or other threshold signing techniques [19, 20]). In a threshold RSA scheme, q is the RSA modulus, K_e is the public key, and the private key K_d is shared by the servers and is used for signing. Secret shares corresponding to K_d are generated by the initial service provider and distributed to the SHs. There are three main protocols in DSO:

1. Service initialization: initializes the share holders, beacons for certain service key;
2. Service provision: provides the desired signing service;
3. Share update: periodically update the secret shares.

Service Initialization This protocol creates multiple shares of the service key K_d and distributes them to randomly selected DSO nodes. The protocol is described in following steps.

- Using Shamir's classical secret sharing scheme [25], the service provider generates a $k-1$ degree sharing polynomial $f(x) = a_0 + a_1 x + ... + a_{k-1} x^{k-1} \pmod q$

where $a_0 = K_d$. It also computes the partial share as $S_i = f(i) \pmod{q}$ for each share holder i, $i = 1, ..., m$. For the purpose of verification of share, we use Feldman's verifiable secret sharing scheme [6], which involves a large prime p such that q divides $p - 1$ and a generator g which is an element of Z_p^* of order q. Also, k public witness of the sharing polynomial's coefficients denoted as $g^{a_0}, ..., g^{a_{k-1}}$ are generated.

- The service provider randomly selects m nodes in the DSO as share holders and then gives every share holder its share S_i together with the k witnesses.
- When the share holders receive their shares they can verify the integrity by checking $g^{S_i} \equiv \prod_{j=0}^{k-1} (g^{a_j})^{i^j}$ (mod p). If it is valid, each SH_i calculates its own partial secret key $d_i = S_i l_i(0)$ (mod q), where $l_i(0)$ is lagrange coefficients and $l_i(0) = \prod_{j=1, j \neq i}^{k} \frac{0-j}{i-j}$ (mod q).
- Each SH replicates its share and partial secret key d_i to other $n_h - 1$ randomly selected server nodes. All these SHs will notify all beacons (whose IDs are identified with hash values of the service key tag) their roles and ask them to store IDs of the SHs for this certain signing service.

Service Provision Once a signing service is initialized, DSO can provide signing service that requires the signing key.

- The client sends service request to DSO, asks for signing message m with the service private key corresponding to a service key tag K_t. The request is forwarded by AP to beacons, then to the proper SHs.
- When the share holders receive the signing request, each SH_i generates its partial signed result (partial signature) $P_i = m^{d_i} \bmod q$ using its partial secret key d_i. These partial results are sent to a beacon (combiner). In order for the beacon to verify the validity of the partial results, we use the following scheme [24]. When SH_i sends P_i, it also sends g^{S_i}, r, c, A_1, A_2. Here $A_1 = g^u, A_2 = m^u$ where u is a random number. $r = u - cS_i$, $c = hash(g^{S_i}, P_i, A_1, A_2)$. All calculations are on mod p.
- When the beacon receives k or more distinct partial results, it first verifies the following three equations (on mod p):

$$g^{S_i} \equiv \prod_{j=0}^{k-1} (g^{a_j})^{i^j};\ g^r (g^{S_i})^c \equiv A_1;\ m^r (P_i)^c \equiv A_2$$

  If these equations hold, then it means that the partial result is valid. After verifying k number of shares, the beacon can generate the final result F.

$$F = \prod_i P_i = \prod_i m^{d_i} = m^{\sum_i d_i} = m^{\sum_i (S_i l_i(0))}\ (mod\ q).$$

  Note $m^{\sum_i (S_i l_i(0))}\ (mod\ q) \neq m^{k_d}$ (mod q). We must apply K-bounded coalition offsetting algorithm [14] to obtain the final signature m^{k_d} (mod q). Note the original K-bounded coalition offsetting algorithm in [14] has a robustness problem, which was pointed in [19, 11] and corrected in [11, 10].

– The final signature is then sent to the AP who forwards it to the client.

Blinding the message: For privacy and confidential reason, the client can use blinding technique to hide the original message from the beacon and other nodes in DSO. Instead of sending the original message m, it chooses a blinding factor b at random and computes $s = b^{K_e}$ (again, all these and following calculations are on mod q) using public key K_e. The client then sends $m \cdot s = mb^{K_e}$ to the overlay, which returns $F' = (m \cdot s)^{K_d} = m^{K_d}b$. Finally the client can remove the blinding factor b by $F = F'/b$. Thus, the client obtains the final signature and does not leak any original information to DSO.

Share Update To enhance security, we use proactive secret sharing scheme [9] to update each share holder's share periodically without reconstructing the service key K_d. In the update procedure, all share holders with the same share copies elect one representative SH to take part in the update protocol. After the update procedure, representatives will replicate the updated share to the other SHs.

– Representative share holder SH_i generates a random update polynomial $f_i(x) = b_{i,0}x + ... + b_{i,k-1}x^{k-1}$ (mod q) with secret part as 0. SH_i computes m subshares $S_{i,j} = f_i(j)$ (mod q) and securely sends it to SH_j, $j = 1, ..., m$.
– After collecting m subshares $S_{i,j}, j = 1, ..., m$, representative SH_i can calculate the new updated share as $S_i^{'} = S_i + \sum_{j=1}^{m} S_{j,i}$ (mod q).

4 Reliability and Security Analysis

In this section, we analyze the fault-tolerance and security of DSO. Table1 list the notations we will use in the following analysis.

Table 1. Some Notations

N_n	total number of nodes in overlay	n_a	number of APs
n_t	number of attacked nodes	k	threshold value of secret sharing
n_b	number of beacons per service	m	number of distinct shares per service
n_h	number of each share's copies	x	total number of services provided by DSO

4.1 Fault-Tolerance Analysis

Fault tolerance of the system is measured as the probability that the service will be available to the clients in spite of the failures of individual nodes. For a given service, $R_{sys}(t)$ represents the probability that the service is available during time interval $(0, t)$. Since there are three components of the service, namely APs, beacons and SHs, all of them need to be operating correctly for the service to be available. Reliability of the system can be written as $R_{sys}(t) =$ Pr(At least one AP operates correctly during $(0, t)$)

$\times$ Pr(At least one beacon operates correctly during $(0, t)$) $\times$ Pr(At least k distinct share holders operate correctly during $(0, t)$).

Every node in the DSO can act as an AP. We are assuming the beacons and share holders for the service are chosen independently. Thus, a given node in the system may have more than one role. To further simplify the computation, we assume that the reliability of every node in the network is equal to $R(t)$. If there are n parallel modules in the system each with reliability $R(t)$ which provide identical service, then the probability that at least one of them will be operating correctly is $R_n(t) = 1 - (1 - R(t))^n$. Reliability of the three components in the service can be represented using the following equations: $R_{AP}(t) = 1 - (1 - R(t))^{n_a}$, $R_{Be}(t) = 1 - (1 - R(t))^{n_b}$, $R_{SH}(t) = 1 - (1 - R(t))^{n_h}$.

Then, we compute the reliability of the system using Eq.(1).

$$\begin{aligned} R_{sys}(t) &= R_{AP}(t) \cdot R_{Be}(t) \cdot \left(\sum_{i=k}^{m} \binom{m}{i} R_{SH}(t)^i (1 - R_{SH}(t))^{n_h(m-i)} \right) \qquad (1) \\ &= \left(1 - (1 - R(t))^{n_a}\right) \left(1 - (1 - R(t))^{n_b}\right) \left(\sum_{i=k}^{m} \binom{m}{i} (1 - (1 - R(t))^{n_h})^i (1 - R(t))^{n_h(m-i)} \right) \end{aligned}$$

Using the reliability equation of the system, it is easy to calculate the mean time to failure as $MTTF = \int_0^\infty R_{sys}(t)$.

Consider an overlay network containing 100 nodes. Suppose we have 10 different services each using $(6, 10)$ to share the secret. Suppose the reliability of a single node is 0.9 during a certain time interval. If these services are deployed in a separate set of nodes with no replication, then the reliability of each service during the time interval is equal to 0.9984 which is two 9s after the decimal point. (For the remainder of this paper, we will write reliability in terms of number of 9s after the decimal point.)

Now, we group $10 \times 10 = 100 = N_n$ nodes together as DSO. We still use the $(6, 10)$ scheme for every service as before. Assume there are ten beacons and each share has four copies, i.e. $n_a = 100, n_b = 10, n_h = 4$. Then according to Eq.(1), the reliability for every service becomes nine 9s, which is much higher than the 2 9s for the services deployed separately.

Note that when using DSO we have much more nodes than required by the service. Many of these overlay nodes are used only for light-weight routing purposes and are not involved in threshold signing or replication. This is not a limitation but a design choice. DSO is designed as a scalable infrastructure to provide a platform for a large number of services. DSO may have a very large overlay and not limited to be within a single organization.

Table 2 shows the reliability of a pure threshold scheme and DSO, given different single node reliability. We can see that for the same threshold parameter $(6, 10)$ DSO has a much higher reliability than the pure threshold scheme. Even when the pure threshold scheme uses $(60, 100)$, which requires 100 servers for a single service, our DSO with following parameters $N_n = 100, n_a = 100, n_h = 4, n_b = 15, k = 8, m = 15$ can still beat the pure threshold scheme when the single node's reliability is not very high. When the single node reliability is very high (greater than 0.9), then both the (60,100) threshold scheme and DSO scheme can achieve high enough reliability (larger than 14 9's). We can see that the pure threshold scheme is very sensitive to the single reliability and the total number of servers. It can only achieve good reliability when using many servers with high single reliability. Whereas DSO can use a small number of servers (here servers mean the nodes really involved in threshold signing or replication) to achieve better performance even when the single reliability is not high.

Table 2. Reliability Comparison ($N_n = 100, n_a = 100, n_h = 4$ in DSO)

Single reliability	0.7	0.8	0.9	0.99		0.7	0.8	0.9	0.99
(6,10) scheme	0.8497	0.9672	0.9984	7 9's	(60,100) scheme	0.9875	5 9's	15 9's	53 9's
DSO ($n_b = 10$)	4 9's	6 9's	9 9's	15 9's	DSO ($n_b = 15$)	7 9's	10 9's	14 9's	15 9's

4.2 Intrusion-Tolerance Analysis

The Threat Model Before security analysis, we first make clear the threat model of DSO. The goal of DSO is to enhance the tolerance to faults and attacks from architecture level. DSO itself is not able to solve all the security problems in the distributed environments. We assume the basic techniques in use, i.e., threshold cryptography, secret sharing and structured P2P routing, are secure. Furthermore we assume that proper authentication, secure communication based on cryptography, traffic analysis prevention and intrusion detection techniques can be used. All these can provide us the following guarantees.

– The overlay network is secure and it prevents the routing attacks.
– The communication between overlay nodes is secure in that authentication and encryption are used between overlay nodes. Also traffic pattern analysis is prevented.
– Every node in the overlay can verify and identify the illegitimate traffic sent to them. Attacker cannot control the node for a long time. Once a node is found under attack/control, it will go off-line and be repaired.
– Attackers do not know the addresses of the beacons and the share holders. Like clients and other routing nodes of overlay, attackers only know the service key tag and its neighbor overlay nodes.

With the above security guarantees, nodes in DSO can still be denied of service or temporarily broken/controlled by the attackers. An attacker can launch three kind of attacks.

1. Availability attack: Attacker can launch DoS attacks on n_t nodes in the network. This may deny the signing service provided by the system to clients.
2. Confidentiality attack: Attacker can attack the nodes to obtain the shares and try to acquire the original service private key (signing key).
3. Integrity attack: Attacker can modify the shares on nodes in the overlay.

In the following sections, we will analyze the intrusion-tolerance of DSO under these three attacks in detail.

Static Intrusion-Tolerance In static attack mode, we assume that there are n_t nodes under attack and there is no recovery.

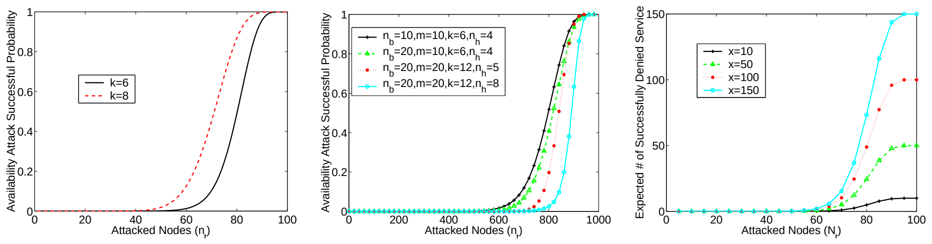
(1) Availability Attack

Suppose we have a set of a nodes and we randomly select b nodes from it. Let $P_h(a, b, c)$ denotes the probability that selected nodes contains a given set of c nodes. Using elementary combinatorics, one can see that $P_h(a, b, c) = \binom{a-c}{b-c}/\binom{a}{b} = \binom{b}{c}/\binom{a}{c}$ when $b \geq c$, and $P_h(a, b, c) = 0$ when $b < c$.

When n_t nodes are attacked, the probability that at least one AP still works is $1 - P_h(N_n, n_t, n_a)$ and the probability that at least one beacon still works is $1 - P_h(N_n, n_t, n_b)$. For each distinct share having n_h copies, the chance of at least one copy available is $1 - P_h(N_n, n_t, n_h)$. A given service will still be available if at least one AP is available, at least one beacon is available and k out of m distinct shares are available. Thus, the probability of successfully denying a given service is

$$Pr_{d1} = 1 - (1 - P_h(N_n, n_t, n_a))(1 - P_h(N_n, n_t, n_b)) \times \left(\sum_{i=k}^{m} \binom{m}{i} (1 - P_h(N_n, n_t, n_h))^i P_h(N_n, n_t, n_h)^{m-i} \right) \tag{2}$$

Figures 2(a) shows a small DSO example. The size of DSO is only 100 ($N_n = 100$). We select all 100 nodes as APs and 10 nodes as beacons. The threshold scheme we use is 6 out of 10. Every share has four replicated copies ($n_h = 4$). When there are $n_t = 10$ nodes attacked, the probability of a successful service availability attack is almost negligible (1.1546×10^{-13}). When attacked nodes increase to 30, the probability comes to be 3.4754×10^{-6}. Even when half of the nodes are attacked, the successful attack probability is still low (0.0013). From the figure we can see that only when the number of attacked nodes is more than 70, the probability increases rapidly. Only after more than 90 nodes are attacked, does the probability become near to 1. Also, given a fixed m, using a larger k will slightly increase the chance of a successful attack. This is because, with a larger k, an attacker needs to attack fewer servers to deny the service.



(a) Small DSO ($N_n = 100$, $n_a = 100$, $n_b = 10$, $m = 10$, $n_h = 4$)

(b) Larger DSO ($N_n = 1,000$, $n_a = 1,000$)

(c) Expected Number of Attacked Service. ($N_n = 100$, $n_a = 100$, $n_b = m = 10$, $k = 6$, $n_h = 4$)

Fig. 2. Availability Attack

In Figure 2(b) we see a larger DSO with $N_n = 1,000$. It is clear that a larger DSO can tolerate more attacks even with other parameters same as Figure 2(a). It is also clear that increasing the number of beacons, SHs and the number of copies of each share (n_h) will somewhat reduce the success probability of an availability attack. However, when less than 600 nodes are attacked, successful attack probabilities are close to zero. This indicates that the size of DSO is a very important factor to enhance the intrusion-tolerance.

We have so far calculated the successful attack probability to a specific service. Consider that the overlay provides x number of services. In this situation when an attacker compromises certain number of nodes, it may deny more than one service. Suppose we are interested in knowing the probability of denying at least one service. Prob(At least one service compromised) $=$ $1 -$ Prob(no service is compromised) $= 1 - \prod_{i=1}^{x}(1 - Pr_{d1})$. The upper bound on this probability is xPr_{d1}. Suppose there are total of 1,000 nodes DSO and a total of 1,000 different services on DSO. For $n_a = 1,000, n_b = 10, m = 10, k = 6, n_h = 4$, if attacker attacks 200 nodes, the probability of denying one or more service is less than $1,000 \times 1.7047 \times 10^{-7} = 1.7047 \times 10^{-4}$. This value is still negligible.

Now we want to compute the probability of denial of exactly y services in DSO. This probability will be a binomial distribution $Pr_d(y) = \binom{x}{y}(Pr_{d1})^y(1 - Pr_{d1})^{x-y}$. We can calculate the expected number of services brought down by the attacker as $\sum_{i=1}^{x} i \cdot Pr_d(i) = xPr_{d1}$.

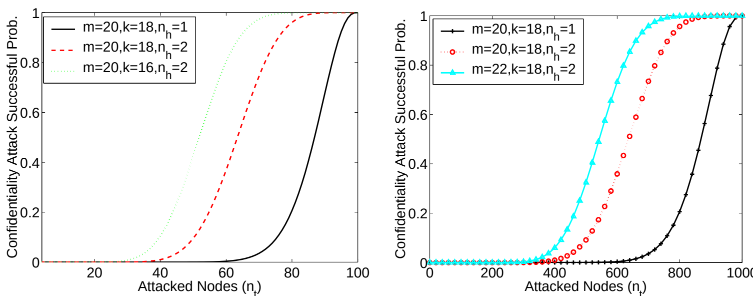
Figure 2(c) plots the expected number of services that are denied given a certain number of attacks. Here we only use a small DSO with small parameters: $N_n = 100, n_a = 100, n_b = m = 10, k = 6, n_h = 4$ (using a larger DSO will obviously improve the performance). The result shows that when the number of attacked nodes is less than 60, almost no services are denied on DSO when the total number of services varies from 10 to 150. For example, when $x = 100$, attacking 30 random nodes can expect to denial 0.00034754 service on DSO. This shows that DSO can provide high availability.

(2) Confidentiality Attack

To analyze the probability of a successful confidentiality attack, Pr_{s1}, we first introduce another new function $P_g(a, b, c)$. Suppose we have a set of a nodes and we randomly select b nodes from it. $P_g(a, b, c)$ denotes the probability that selected nodes do not contain any of the nodes from a given set of c nodes. Evidently, $P_g(a, b, c) = \binom{a-c}{b}/\binom{a}{b}$ when $b \le a - c$, $P_g(a, b, c) = 0$ when $b > a - c$.

A confidentiality attack may succeed only when an attacker can successfully get at least k distinct shares for a certain service. Since the attacker has no knowledge of where the share holders are located in the network, he needs to randomly attack a large set (n_t) of nodes on DSO. The probability of successfully stealing given service secret key Pr_{s1} is that of at least k out of m distinct shares be stolen. For each distinct share, $P_g(N_n, n_t, n_h)$ means no copy of this share is stolen while $1 - P_g(N_n, n_t, n_h)$ indicates that at least one copy is stolen. The probability of a successful confidentiality attack can be computed as

$$Pr_{s1} = \sum_{i=k}^{m} \binom{m}{i} (1 - P_g(N_n, n_t, n_h))^i P_g(N_n, n_t, n_h)^{m-i} \quad (3)$$



(a) Small DSO ($N_n = 100$) (b) Larger DSO ($N_n = 1,000$)

Fig. 3. Confidentiality Attack

Figure 3(a) shows a small DSO example. Here $N_n = 100, n_a = 100, n_b = 10, m = 20, n_h = 2$. Assume $k = 18$, when attacker attacks $n_t = 20$ nodes, the chance of success is negligible (9.1880×10^{-7}). Even when half of nodes are attacked, the chance of a successful confidentiality attack is still low (0.0954). We can see from the figure that the confidentiality attack is very sensitive to the number of shares in DSO. Smaller n_h will achieve better attack tolerance than larger n_h. This makes sense because more copies of shares mean higher chance to leaking the shares. Given a fixed m, using a larger k will also enhance intrusion-tolerance. This is obvious because attackers need to attack more nodes in order to acquire at least k distinct shares.

In Figures 3(b), we see a larger DSO with 1,000 server nodes. For a fixed k, increasing m or increasing n_h will both enhance the attack probability because increasing m or n_h increases the number of shares in the system. Thus the attacker has more chance of getting k shares. For $m = 20, k = 18, n_h = 1$, when less than about 600 node are attacked, the probability of the success of attack is nearly zero.

The analysis of the probability of acquiring at least one service key, acquiring y service secret key and the expected number of service secret keys attacker can steal is similar to the analysis of that of availability attack.

We use different parameters in availability and confidential attack tolerance analysis. This is because availability and confidentiality has contradicting requirements. For example, larger n_h will achieve high availability attack tolerance but low confidentiality attack tolerance. As we can see there is some trade-off between availability and confidentiality. We will present some trade-off techniques in Section 5.1.

(3) Integrity Attack

One aspect of the integrity attack aims to corrupt enough shares so that attacker can forge the secret signing key. In this case the intrusion-tolerance against integrity attacks is the same as that against confidentiality attacks because when a node is attacked its share can either be disclosed or corrupted. Once an attacker has modified k copies of distinct shares, a client might receive the corrupted signature when beacon contacts those k nodes with corrupted shares.

Another aspect of the integrity attack is that the attacker modifies all copies of $n - k + 1$ distinct shares. Now no client will ever be able to use the signing service. This integrity attack is similar to Denial of Service. The successful attack probability is the same as that of availability attacks analyzed before.

As both aspects of the integrity attack are covered by the availability and confidentiality analysis, we will not further discuss this attack separately in this paper.

Dynamic Intrusion-Tolerance In a dynamic situation, besides considering the attacks, we also consider the recovery of nodes. Out of n_t nodes attacked by the attacker, $n_t - i$ nodes are detected by the intrusion detection module and are already under recovery. The remaining i nodes are still compromised and not detected. Nodes under recovery are removed from the system. Let π_i be the steady-state proportion of time that i number of nodes are under attack and not removed from DSO. When i out of the n_t nodes are under attack and $n_t - i$ are under recovery, there are $N_n + i - n_t$ active nodes on DSO. Now we extend the static case above to the dynamic cases. In a steady state, the probability of a successful availability attack is computed in Eq.(4) where $P_h(a, b, c)$ is equal to $P_h(a, b, a)$ when $c > a$. Eq.(5) gives the probability of successfully stealing a given service secret key.

$$\begin{aligned} Pr_{d1} = \textstyle\sum_{i=0}^{n_t} \pi_i [1 - (1 - P_h(N_n + i - n_t, i, n_a))(1 - P_h(N_n + i - n_t, i, n_b)) \times \\ \left(\textstyle\sum_{j=k}^{m} \binom{m}{i} (1 - P_h(N_n + i - n_t, i, n_h))^j P_h(N_n + i - n_t, i, n_h)^{m-j} \right)] \end{aligned} \tag{4}$$

$$Pr_{s1} = \sum_{i=0}^{n_t} \pi_i \left(\sum_{j=k}^{m} \binom{m}{i} (1 - P_g(N_n + i - n_t, i, n_h))^i P_g(N_n + i - n_t, i, n_h)^{m-i} \right) \tag{5}$$

We need to calculate π_i to determine the probabilities. We consider two kinds of attack processes (i.e. serial attack and parallel attack) and repair processes (i.e. serial repair and parallel attack). In serial process, nodes can only be attacked or repaired one by one. In parallel process, multiple nodes may be simultaneously attacked or repaired. Attack delay time is the difference between the time when an attacked node is removed from the overlay and when the attacker realizes the attacked node is removed and redirects the attack to a new node in the overlay. Repair delay time is the difference between the time when a node is first attacked and when DSO detects the attack and removes the node from the overlay. We assume that attack delay time and repair delay time follow exponentially distributed random variables with respective rates of λ and μ. Assume $\rho = \lambda/\mu$. We can model the system as a birth-and-death exponential queueing system using Markov model. According to different attack and repair processes we have four Markov models in the Figure 4. Every state i in the figure represents the state that exactly i nodes are under attack at the time. In the case of a serial attack, the rate of transition form state i to state $i + 1$ is λ, while in the parallel case, the rate is $(n_t - i)\lambda$. When the repair is serial, the rate of transition from state i to state $i - 1$ is μ, while in the case of parallel repair the rate is $i\mu$.

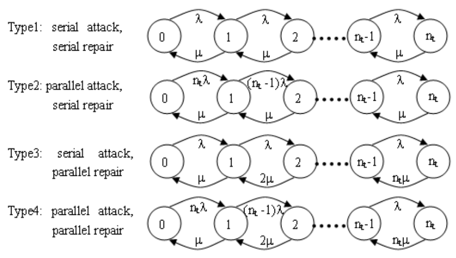
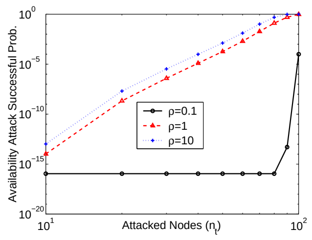

Type1: serial attack, serial repair
Type2: parallel attack, serial repair
Type3: serial attack, parallel repair
Type4: parallel attack, parallel repair
0
1
2
n_t-1
n_t
λ
μ
n_tλ
(n_t -1)λ
2μ
n_tμ

Fig. 4. Dynamic Attack and Repair Models

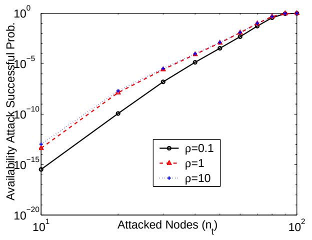
According to the balance equation [22], we can compute π_i using the general formula Eq.(6). Details on analysis for every situation can be found in [13]. We have:

$$\pi_0 = \frac{1}{1+\sum_{n=1}^{n_t} \frac{\lambda_0 \cdots \lambda_{n-1}}{\mu_1 \cdots \mu_n}}; \pi_i = \frac{\frac{\lambda_0 \cdots \lambda_{i-1}}{\mu_1 \cdots \mu_i}}{1+\sum_{n=1}^{n_t} \frac{\lambda_0 \cdots \lambda_{n-1}}{\mu_1 \cdots \mu_n}} \text{ when } i > 0 \qquad (6)$$

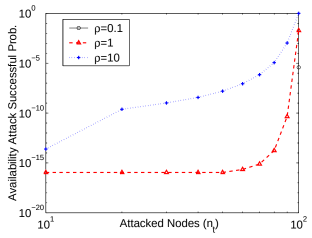
here λ_i means the attack rate of transition from state i to $i+1$, μ_i means the repair rate of transition from state i to $i-1$.



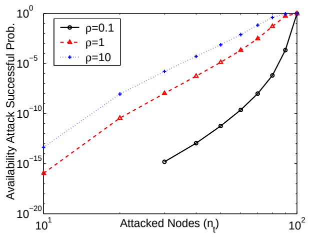
(a) Type1: Serial Attack, Serial Repair



(b) Type2: Parallel Attack, Serial Repair



(c) Type3: Serial Attack, Parallel Repair



(d) Type4: Parallel Attack, Parallel repair

Fig. 5. Dynamic Case for Availability Attack ($N_n = 100, n_a = 100, n_b = m = 10, k = 6, n_h = 4$)

We plot Pr_{d1} under four conditions in Figure 5. The plot of ρ vs Pr_{s1} is similar to the plot of Pr_{d1}. From Figure 5 we can see that with a fixed ρ (e.g. $\rho = 1$), in terms of intrusion-tolerance, $Type2 < Type1 < Type4 < Type3$. Serial attack and parallel repair achieves the best performance of intrusion-tolerance. It is understandable because this case is the best case we can have: the highest repair rate and the lowest attack rate. Parallel attack and serial recovery is the worst case with the highest attack rate and the lowest repair rate. Not surprisingly with a larger ρ, the intrusion-tolerance will be worse because a larger ρ means a higher attack rate compared to recovery rate.

5 Discussion

Availability and confidentiality have conflicting requirements. If we try to improve one by changing parameters like k, m, n_h then the other will suffer and vice versa. We would like to find a trade-off between these two depending on the relative importance of availability and confidentiality ($A = lC$). Availability and confidentiality of the system can be measured as $(1 - Pr_{d1})$ and $(1 - Pr_{s1})$ respectively in DSO. If the desired ratio of availability to confidentiality is l, then $(1 - Pr_{d1}) = l(1 - Pr_{s1})$. Given l and some of the parameters, one can set the proper values of the rest of the parameters.

5.1 Intrusion-Tolerance Comparison

Other popular schemes which provide similar functionalities as DSO are pure replication, pure threshold scheme, threshold scheme plus replication, threshold scheme plus quorum. In pure replication, the key is replicated to all m_r servers. Pure threshold scheme uses (k_s, m_s) scheme where key is divided into m_s shares and at least k_s shares are required to reconstruct the secret. As the name suggests, threshold scheme plus share replication creates n_h replications of each m_{sr} shares.

In all these schemes, an attacker may know the servers providing the service and attack them. The attacker can easily determine the set of minimum servers it needs to attack to bring the system down. If the number of attacked nodes is below this threshold then the chance of successful attack is zero. If an attacker compromises more servers than the threshold, it will succeed with probability one. Table 3 lists the comparison of all existing schemes in terms of availability and confidentiality.

Table 3. Intrusion-tolerance Comparison (Attack Successful Probability)

	Availability Attack	Confidentiality Attack
Pure Replication	$\begin{cases} 0 & n_t < m_r \\ 1 & n_t \geq m_r \end{cases}$	$\begin{cases} 0 & n_t < 1 \\ 1 & n_t \geq 1 \end{cases}$
Pure threshold scheme	$\begin{cases} 0 & n_t < m_s - k_s + 1 \\ 1 & n_t \geq m_s - k_s + 1 \end{cases}$	$\begin{cases} 0 & n_t < k_s \\ 1 & n_t \geq k_s \end{cases}$
Threshold scheme+Replication	$\begin{cases} 0 & n_t < (m_{sr} - k_{sr} + 1)n_h \\ 1 & n_t \geq (m_{sr} - k_{sr} + 1)n_h \end{cases}$	$\begin{cases} 0 & n_t < k_{sr} \\ 1 & n_t \geq k_{sr} \end{cases}$
DSO	Eq. (2)	Eq. (3)

Let's compare the different schemes using a specific example. In order to have a fair comparison between other replication or threshold schemes and DSO, we need to use the same number of original servers. It will not be fair to compare a (50,100) pure threshold scheme and a 100 node DSO with a (6,10) scheme with no replication. Even though DSO has a network size of 100, only 10 servers are being used for this particular service. All other nodes are just part of the network. This case is similar to any distributed service provided on top of WAN or the Internet. We do not count routers or hosts in the path of routing as a part of servers. Thus, for all the schemes we compare, we assume that the total number of servers is 10. For DSO, we assume the overlay has 100 nodes.

Let's examine the intrusion-tolerance of each scheme. All schemes except DSO cannot get high intrusion-tolerance under both attacks. We use the importance comparison equation $A = lC$ to optimize the parameters for each scheme. To simplify matters, we will choose $l = 1$ which means attack-tolerance for availability and confidentiality is equally important. For other schemes, importance can be measured using the threshold.

For the pure replication scheme, threshold for availability attack-tolerance is m_r. But confidentiality is compromised even if a single server is successfully attacked.

For pure threshold scheme, we have the equation $m_s - k_s + 1 = k_s$. From this equation we get $k_s = (m_s + 1)/2 = 11/2 \approx 5$ which is the optimized value in order to achieve both attack-tolerance. So we take $(5, 10)$ for pure threshold scheme.

Threshold plus replication scheme uses (k_{sr}, m_{sr}) secret sharing scheme and each share has n_h copies. Thus, the total number of servers in the system is $n_{sr} = m_{sr} \times k_{sr}$. By plugging in the values of availability and confidentiality in the importance equation we have $(m_{sr} - k_{sr} + 1)n_h = k_{sr}$. Thus $k_{sr} = (n_{sr} + n_h)/(n_h + 1) = 1 + (n_{sr} - 1)/(n_h + 1)$. This indicates that k_{sr} increases as n_h decreases. For $n_h = 2$, $k_{sr} = 12/3 = 4$. For $n_h = 1$ this becomes pure threshold scheme. For this scheme we take the optimized parameters as $k_{sr} = 4, m_{sr} = 5, n_h = 2$.

We place 10 servers into an overlay with 100 nodes. Note that we still use only 10 nodes as SHs for threshold signing or replication. Other nodes are just overlay nodes in charge of P2P routing. For these 10 servers, we use two strategies: one is a (5,10) threshold scheme ($k = 5, m = 10, n_h = 1$), the other is a (4,5) threshold scheme plus each share has 2 replications ($k = 4, m = 5, n_h = 2$).

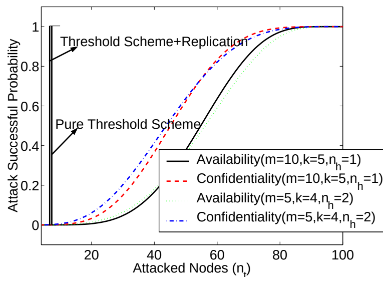


Fig. 6. Intrusion-tolerance Comparison of Three Schemes

Figure 6 plots the intrusion-tolerance of three schemes (pure threshold scheme, threshold scheme plus replication, DSO) given the same number of total servers $n = 10$. It is obvious that DSO has the best intrusion-tolerance. The probability of intrusion-tolerance is still high even when there are many overlay nodes attacked.

From Figure 6, we can see that a (5,10) scheme on the top of DSO will have a very similar attack-tolerance to a (50,100) pure threshold scheme. But using a (5,10) scheme on the top of DSO is better because it has less overhead, in addition to use less real servers. In the case of DSO, the combiner (beacon) needs to contact just 5 to 10 servers to obtain the shares. Whereas in the (50,100) scheme, the combiner needs to contact at least 50 servers. Also, verifying and combining 50 shares will take more time than verifying and combining just 5 shares.

5.2 Performance Evaluation

There are two types of overhead in the system, computation and communication. Distributed signing requires partial signing at SHs and combining at the combiner. The cost of one partial signing operation is the same as the signing using a complete RSA private key. Combining of data takes more time than partial signing. Time taken to partially sign the certificate and combine the partial signature by the combiner is proportional to the number of different shares. In DSO architecture, we do not need to use very high values of k and m to provide high security. As we have a smaller number of shares, time to sign or combine the shares is not very large. But in other system like pure threshold scheme one needs to have a large k and m to provide high security. Thus, these schemes require more time to combine the shares.

"MessageHop" is used as a metric to evaluate the communication cost. MessageHop is defined as the number of overlay nodes covered by the packet to reach the destination. The communication cost in DSO is the routing cost from AP to a beacon and then to m_{DSO} distinct SHs. The average number of hops for DHT-based routing to any destination is $log(N_n)/2$. So the total cost for sending request to share holders is $log(N_n)/2 \times (1 + m_{DSO})$. $log(N_n)$ can range from 5 to 30 depending on the size of the overlay. Although this involves some communication cost, it is still reasonable considering the fault- and intrusion-tolerance benefits provided by the overlay.

6 Conclusion

In this paper, we have discussed the importance of as well as the challenges in building dependable systems with both high fault-tolerance and intrusion-tolerance. Using digital signing service as a motivated example, we proposed a novel architecture, Dependable Signing Overlay (DSO). The fault-tolerance feature is achieved via using the threshold scheme plus replication. The intrusion-tolerance feature is obtained via using the threshold scheme plus anonymous servers, so the attackers cannot know which servers to attack in order to deny signing service or steal/corrupt signing key. We designed DSO as a P2P overlay server network and adopted the techniques of structured P2P overlay routing based on DHT. We derived analytical models and presented reliability (fault-tolerance) and security (intrusion-tolerance) analysis. Our results show that DSO provides very high fault-tolerance and intrusion-tolerance. We also compared DSO with other existing techniques. Our results showed that DSO provides high fault-tolerance and much higher intrusion-tolerance.

In conclusion, we believe that DSO is a promising and scalable platform to build dependable signing services.

References

1. Ross J. Anderson. The eternity service. http://www.cl.cam.ac.uk/users/rja14/eternity/eternity.html.
2. C. Cachin and J. Poritz. Secure intrusion tolerant replication on the internet. In *DSN 2002*, 2002.
3. Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Security for peer-to-peer routing overlays. In *OSDI '02*, December 2002.
4. B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of the 26th IEEE Symposium on the Foundations of Computer Science*, 1985.
5. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Crypto'89*, 1989.
6. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of IEEE 28th Annual Symposium on the Foundations of Computer Science*, October 1987.
7. The free haven project. http://www.freehaven.net/.
8. Maurice Herlihy. A quorum-consensus replication method for abstract data types. *ACM Trans. Comput. Syst.*, 4(1):32–53, 1986.
9. A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing, or: How to cope with perpetual leakage. In *Proceedings of CRYPTO'95*, 1995.
10. Stanislaw Jarecki and Nitesh Saxena. Futher simplifications in proactive rsa signature schemes. In *TCC'05*, 2005.
11. Stanislaw Jarecki, Nitesh Saxena, and Jeong Hyun Yi. An attack on the proactive rsa signature scheme in the ursa ad hoc network access control protocol. In *SASN '04*, pages 1–9, New York, NY, USA, 2004. ACM Press.
12. Angelos Keromytis, Vishal Misra, and Dan Rubenstein. SOS: Secure Overlay Services. In *ACM SIGCOMM'02*, August 2002.
13. L. Kleinrock. *Queueing Systems,Volume I: Theory*. 1975.
14. J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *ICNP'01*, November 2001.
15. Cachin K. Kursawe, A. Lysyanskaya, and R. Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In *ACM CCS'02*, 2002.
16. Subramanian Lakshmanan, Mustaque Ahamad, and H. Venkateswaran. Responsive security for stored data. In *ICDCS'03*, 2003.
17. H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang. Self-securing ad hoc wireless networks. In *ISCC'02*, July 2002.
18. D. Malkhi and M. Reiter. Byzantine quorum systems. *Distributed Computing*, 11(4), 1998.
19. Maithili Narasimha, Gene Tsudik, and Jeong Hyun Yi. On the utility of distributed cryptography in p2p and manets: The case of membership control. In *ICNP '03*, 2003.
20. Maithili Narasimha, Gene Tsudik, and Jeong Hyun Yi. Efficient node admission for short-lived mobile ad hoc networks. In *ICNP '05*, 2005.
21. T. Rabin. A simplified approach to threshold and proactive rsa. In *Crypto'98*, 1998.
22. Sheldon M. Ross. *Probability Models for Computer Science*. 2002.
23. A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Middleware'01*, 2001.

24. B. Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Crypto'99*, 1999.
25. A. Shamir. How to share a secret. *Communications of ACM*, 24(11), 1979.
26. V. Shoup. Practical threshold signatures. In *Proceedings of EUROCRPT'00*, 2000.
27. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM'01*, 2001.
28. Marc Waldman, Aviel D. Rubin, and Lorrie Faith Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proc. 9th USENIX Security Symposium*, pages 59–72, August 2000.
29. T. Wu, M. Malkin, and D. Boneh. Building intrusion tolerant applications. In *Proceedings of 8th USENIX Security Symp (Security'99)*, 1999.
30. L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Networks*, 13(6), 1999.
31. L. Zhou, F.B. Schneider, and R. Van Renesse. A secure distributed online certification authority. *ACM Transactions on Computer Systems*, 20(4), 2002.