

# Secure Communication in a Distributed System Using Identity Based Encryption

Tyron Stading  
IBM, Austin, Texas 78758, USA  
tjstadin@us.ibm.com

## Abstract

*Distributed systems require the ability to communicate securely with other computers in the network. To accomplish this, most systems use key management schemes that require prior knowledge of public keys associated with critical nodes. In large, dynamic, anonymous systems, this key sharing method is not viable. Scribe is a method for efficient key management inside a distributed system that uses identity based encryption (IBE). Public resources in a network are addressable by unique identifiers. Using this identifier as a public key, other entities are able to securely access that resource. We evaluate key distribution schemes inside Scribe and provide recommendations for practical implementation to allow for secure, efficient, authenticated communication inside a distributed system.*

## 1 Introduction

Current methods for securing communications inside distributed systems are impractical. As more distributed systems are built upon public, untrusted networks, securing and authenticating internal communications becomes imperative to maintaining overall integrity and security. Most systems rely on SSL or TLS protocols for secure communication. However, these protocols require the use of asymmetric cryptography keys which make conventional key management schemes impractical for large systems. A better method for key management is needed to practically and efficiently secure internal communication.

Conventional key management schemes inside distributed systems are inefficient for many reasons. In order to identify each node, these systems assume that each node's public key has been signed by a trusted Certificate Authority. This requirement adds considerable cost and complexity for computers wishing to participate in the system. Assuming all nodes have obtained these certificates, most schemes additionally assume prior knowledge of another node's certificates. That is, a node must already know the destination's

certificate before it sends a message. Caching of these certificates introduces problems with trust and storage, requiring every node to locally store a certificate for future communication. For large systems, this adds considerable overhead on local storage. Furthermore, if a distributed system's membership is highly dynamic with nodes constantly joining and leaving the network, many of the certificates can quickly become invalidated. Therefore, current key management schemes are not adequately tailored for secure communication in distributed network.

In this paper we describe the architecture for Scribe and identify its functional components. Scribe is a method for efficient key management inside a distributed system that uses identity based encryption (IBE). Public resources in a network are addressable by unique identifiers. Using this identifier as a public key, other entities are able to securely access that resource. Scribe provides a key escrow service that authenticates nodes based on their public identity and delivers private keys accordingly. This system allows secure, efficient, authenticated communication inside a distributed system.

For the design and analysis of Scribe, we make several assumptions. First, we assume the distributed system is organized by an overlay which allows a particular node to be addressed by the resource or the function the node serves inside the system. Scribe was originally designed for Chord [12] but could also be incorporated into other distributed overlay systems. Furthermore, this paper focuses on the encryption and security of internal communications, and user specified encryption of files is not defined. We also assume an honest-but-curious adversary model in which all nodes correctly perform operations but are interested in compromising the master-key to decrypt communications.

The rest of the paper presents Scribe in the context of a peer-to-peer distributed system. Section 2 presents key technologies used to in the design of Scribe. Section 3 then reviews the problems of securing a master key inside a distributed system and the design choices to address them. In Section 4, we detail the key management scheme that uses a distributed key escrow service for internal communication. Section 5 presents an evaluation of the key distribution

scheme and highlights recommendations for practical implementation. Finally, we present related and future work.

## 2 Background

Scribe provides a key escrow service to all the nodes in a distributed system. Given an identity and correct authentication, Scribe will generate and deliver a private key associated with that node's identity. However, identity is defined by an arbitrary string describing the unique role a node plays in the distributed system. To provide a key escrow service based on these identities, we employ identity based encryption (IBE).

### 2.1 Identity Based Encryption

Identity Based Encryption (IBE) is an encryption scheme where a node's public key is based on an arbitrary string. Shamir [11] originally proposed IBE to simplify certificate management in email systems. For example, if Alice wished to send Bob an encrypted email, there is no need for Alice to obtain Bob's public key certificate. Instead, Alice would be able to encrypt a message to Bob using only his email address as the public key. When Bob receives the email, he authenticates himself to a third party called a Private Key Generator (PKG) and obtains the private key. Bob can then read the email (even if he didn't previously establish a private key).

IBE remained an unsolved problem until recently. Previous solutions had certain limitations. Some solutions specified that users not collude. Others required the PKG to spend a significant amount of time generating each private key generation request. In a recent paper [3], Boneh et.al. describe an IBE system using the Weil Pairing that solved many of the inefficiencies of previous systems. This cryptosystem has chosen cipher text security in the random oracle model assuming an elliptic curve variant of the computational Diffie-Hellman problem. Additionally, the performance of this system is comparable to the performance of ElGamal encryption. Therefore, the Boneh solution is the first practical IBE scheme.

There are four algorithms in an IBE scheme:

**Setup** This function generates global system parameters and generates a master-key.

**Extract** This call uses the master-key to generate a private key that corresponds to an arbitrary public key string ID.

**Encrypt** Takes a message and encrypts it using the public key ID.

**Decrypt** Decrypts a message using the corresponding private key.

By using IBE, nodes can: 1) send messages to recipients who have not yet setup a public key, 2) improve performance and reduce complexity by elimi-

nating the lookup of a recipient's public key, 3) send messages that can only be read at some certain time in the future, 4) proactively refresh a recipient's private key every predetermined period of time, and 5) verify the identity of another node by the role it plays in the distributed system. These features make messaging inside a distributed system secure and efficient. A significant weakness of the key escrow system is the exposure of the master key. By obtaining the master key, any node can encrypt and decrypt a message for any destination. Therefore, securing the master key is crucial. To address this problem, we use threshold cryptography.

### 2.2 Threshold Cryptography

Threshold cryptography [5] addresses this secret key storage problem by breaking a master key into numerous secret shares and storing them on multiple machines. For example, key  $K$  can be broken into  $k1$ ,  $k2$ , and  $k3$ , each of which is stored on a different machine. All components of the broken key must be obtained to reconstruct the entire key. Operations using the key never allow it to be reconstructed in a single location. Master key operations occur by submitting a request to each machine holding a share. The requestor receives all the computed responses and mathematically reconstructs them into the correct message without revealing the master key. For the key to be compromised, a person would have to compromise each node containing the shares of the key. With a large number of key components distributed across the network, this becomes extremely difficult.

Until recently, threshold cryptography suffered from the requirement that the master key be generated by a trusted party. That party would then separate the master key and distribute the components. This would render threshold cryptography useless in a self-managed, distributed system. However, it has been shown that a master key can be generated in a distributed environment without constructing the key at any single node [2]. As long as less than half of the nodes chosen to generate the key do not collude, the key is not compromised during generation.

Scribe uses threshold cryptography to enable secure operations of the key escrow. By secretly storing the master key, a malicious node would have to discover the locations of each of the shares, compromise those machines, and reconstruct the master key. By designing the system to protect against this type of attack, Scribe is able to defend against master key theft and offer a reliable key escrow service.

## 3 Design

This section identifies the problems of storing a master key inside a self-managed distributed system.

Scribe relies on the integrity of the master key to guarantee secure communications. If the key is obtained by a malicious node, messaging is considered to be insecure as all communications can be read by that node. Securing the master key is especially difficult inside a distributed system built upon public, untrusted networks. Because there are no trusted entities, the master key must be protected from generation to storage by a collection of nodes. Scribe addresses these problems by using algorithms for anonymity, bootstrapping, and proactive secret sharing. The security and performance of these algorithms is beyond the scope of this paper.

### 3.1 Anonymity

Anonymity is especially important in the design of Scribe to improve security. By limiting the knowledge of the network topology, a malicious node is unable to effectively identify and attack servers directly. Without anonymity, a malicious node will want to attack those servers holding the components of the master key. An attacker would either want to assume responsibility for that location and inherit the secret key, or identify the IP address of the server and launch an attack to steal the piece of the key. Anonymity makes it very difficult for malicious nodes to perform either of these actions.

Several projects have researched anonymity in distributed hash tables [4, 6] and require certain modifications to the system. We identify two common techniques used to provide anonymity. First, normal `join` operations require a new node to pick a random point in the hash space to join the system. However, a malicious node could repeatedly join and leave the group to obtain information about the overlay. To stop this attack, the `join` operation needs to be altered to only allow a node to join using the area that corresponds to the hash of its IP address, or at least the hash of its subnet address. By limiting the number of locations a node can join, only minimal information can be discovered about the overlay. The second overlay change in Scribe requires the anonymous routing of messages for internal communication. A node should only be aware of its immediate neighbors or a reasonably finite set. Once a message has reached its destination, a reply will follow a similar routing path back to the requestor.

By providing anonymity, Scribe limits information available to malicious entities in order to protect servers storing pieces of the master key.

### 3.2 Bootstrapping

Scribe must achieve critical mass before normal operations can begin. If a master key is generated with only a small number of nodes in the overlay, a malicious node can easily attack and compromise the

key. Therefore, a delayed booting process must be employed.

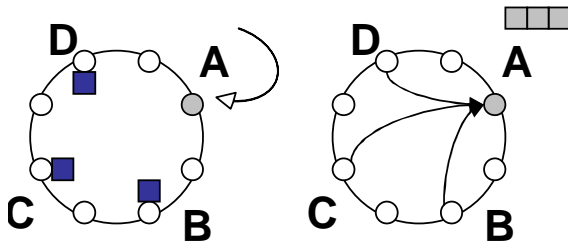
Scribe is able to bootstrap operations in the system once the overlay has achieved critical mass (i.e. enough nodes exist to effectively hide the components of the master key). The system is able to determine critical mass has been reached when the hash space owned by a given node becomes less than a specified fraction of the whole namespace. Since the distribution of joins should be randomly uniform over the system, Scribe is able to trigger a bootstrap whenever any node has taken a sufficiently small area of ownership. Once this occurs, the triggering node randomly picks  $n$  points in the coordinate space and instructs them to generate the components of a master key. To prohibit premature key generation, neighbors of the triggering node validate its area of ownership before forwarding the request. Similarly, a node instructed to generate a component of the key can independently determine if Scribe has critical mass and decides whether the operation should execute. After validating the request, Scribe follows the key generation process described in the next section.

### 3.3 Key (Re)Generation

Key generation is an important component of the system. Previously, threshold cryptography required a trusted dealer to generate a master key, separate it into components, and distribute the pieces to various servers. However, this would not be useful in Scribe as it a) requires a trusted party, and b) constructs the master key in a single location. Using the method described by Boneh [2], we are able to jointly generate an RSA key pair. Through a series of steps, Scribe selects candidates (each picks secret integers), computes  $N$ , performs a biprimality test, and eventually generates a public key. The master key is distributed among many parties and is unknown if less than half of the parties collude. Using this technique, Scribe is able to securely generate a master and public key in an untrusted environment.

A node can trigger the key generation process at various times during the operation of Scribe. Once triggered, the node randomly picks the locations of the key components. Each location is sent a request to generate a component, as well as the anonymous location of all the other servers being used to create a master key. After the key has been generated, the public key is replicated throughout Scribe so each node has a copy of the key.

There are two times when a new key should be generated. The first occurs when the system has reached critical mass. The second triggers when Scribe's master key expires. When this happens, the first server attempting to issue a private key after the expiration date messages the master key component directory (a node that directs requests to shares of the master key)



**Figure 1. As node A joins the system, a request is forwarded for its private keys. Each server holding a part of the master key (B, C, D) computes its portion of the private key and routes that piece back to node A.**

to block all subsequent requests for a short period of time. After the key is regenerated, normal operation resumes. If a regeneration request is received before the master key expires, the request is ignored.

### 3.4 Key Distribution

To provide redundancy, security, and load balancing, we support proactive secret sharing, which provides multiple master key components that can be changed dynamically. After initial generation, the existing servers holding a piece of the master key create additional pieces in a method called *refresh*. The idea is to create new master key component sets so the same sets are not overwhelmed with the requests. This *refresh* method is also useful to redistribute the keys if the system believes one of the servers is compromised. In doing so, the system is able to effectively nullify any effort the malicious node had made toward compromising the master key.

For master key operations, each server holding a piece of the master key knows how to find the other servers in its set, though it cannot determine which IP addresses hold them. Through the distributed hash table, we also specify a certain location as a component set directory for initial discovery of the master key pieces. This allows master key operations to execute more efficiently.

The design of Scribe allows us to ensure that a master key is never reconstructed by the system, a malicious node has no knowledge of the real location of a server, component sets are dynamically recreated in the face of intrusions, and a malicious node must break into all  $n$  servers in a component set to obtain the master key. These features make our system difficult for someone to discover Scribe's secret key.

## 4 Key Management

Once the master key has been securely stored in the distributed system, Scribe can perform the operations of the key escrow service. Each node is responsible for obtaining the private key(s) associated with its public identifier. When a request is submitted, a node messages all the servers (via their identity in the overlay) holding a piece of the master key for the private key associated with its public identifier. Once identified, the requesting node receives the private key and can decrypt messages. This process is depicted in figure 1.

For functional schemes, Scribe must address how nodes are identified, how they are authenticated and given the private key, how other nodes encrypt information for them, and how dynamic membership is handled.

### 4.1 Public Identities

Distributed hash table (DHT) overlays complicate the notion of public identities. A DHT must provide a way to uniquely store all possible documents, so it uses a secure hashing algorithm to uniquely identify a given key. Chord uses the SHA-1 algorithm which produces a 160-bit hash value. Nodes are identified in Chord by assigning it ownership of portions of the namespace. Therefore, a node is publicly identified by what region of the 160-bit hash space it owns and can be messaged by encrypting information to a point within its realm of ownership. However, each node can have billions of identifiable points that can be messaged. A consistent scheme for public identification is needed.

To solve this, we specify that only a portion of the 160-bits be used as public identifiers. For example, a node  $A$  wants to locate a document  $123$ . Node  $A$  hashes  $123$  with SHA-1, receives a 160-bit hash, and uses the  $N$  most significant bits (a system parameter) as the public identifier. The node at the receiving end decrypts the message with the private key corresponding to the  $N$ -bit identifier. Therefore, a secure message was able to be encrypted and decrypted.

However, the number of the most significant bits (MSB) used directly corresponds to the addressable size of the system. If only the first bit was used, only two regions would be uniquely identifiable. Therefore, the number of bits should be chosen according to the expected membership of the network. Table 4.1 illustrates the relationship between the MSB and the addressable regions for encryption.

### 4.2 Authentication

Each node authenticates itself in Scribe by proving the role it plays in the overlay. With Chord, each node is asserting they are responsible for a given region in the namespace. We are able to verify this by leveraging the routing of messages. After a node requests

<i>Num of Most Significant Bits</i>	<i>Total Keys</i>
12	4096
14	16384
16	65536
18	262144
20	1048576

**Table 1. Addressable public key identities based on MSB of 160-bit SHA-1 Hash. The number of keys relate to the expected maximum size of the network.**

a private key based on its MSB identifier, each server holding a part of the master key routes its computed response to the node responsible for the MSB id. The private key can only be reconstructed by compiling all the responses from the servers holding the master key components. Because the overlay will deliver all the pieces of the private key back to the node owning that space, we have verified a node holds that public ID. A denial of service attack against obtaining a key is outside the scope of this paper.

### 4.3 Expiration

Private keys are given an expiration period to improve security. Because nodes can be compromised or rapidly change roles in the overlay, it is important that a private key be given a maximum life span. After a key expires, the node must reauthenticate to obtain a new private key. Doing so maintains the correct association between public identities and private keys.

IBE accomplishes expiration by specifying a valid time segment during which a public key is valid. For example, the public key would consist of the public identifier and a valid time, such as the morning of Nov 12, GMT 2002. Scribe does not specify the expiration period as the performance tradeoff is dependent on the security of the system.

### 4.4 Encryption

The specific format of a node’s public key is comprised of three parts. First, each node in Scribe must have the system’s public key  $K$  generated after achieving critical mass. The node also needs the MSB id of the node to message as well as the valid period for the message. The exact syntax of a node’s public key would be:

public key  $K$  || MSB ID || Valid Timestamp

This string is then used as a public key to encrypt a message payload to be decrypted by a specific node.

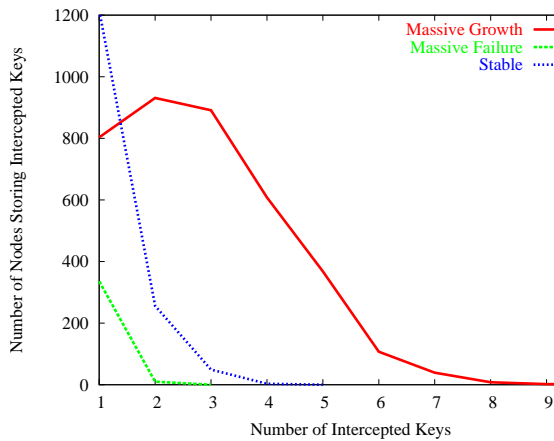
## 5 Evaluation

In our preliminary evaluation of Scribe we focused on the security and dissemination of private keys to nodes. Specifically, we looked at how dynamic overlay systems can allocate and share keys while still maintaining security. In a large, static system, Scribe is secure in that private keys are rarely shared. However, overlay systems usually share responsibilities between nodes. As new nodes join an overlay system, they assume the responsibility and keys for the region they occupy. However, the previous node that occupied that space also holds a copy of the private keys given to the joining node. The previous node now has the ability to intercept and decrypt messages intended for another node, decreasing the security of Scribe. Therefore, we analyzed Scribe in the context of sharing keys inside a large dynamic anonymous network.

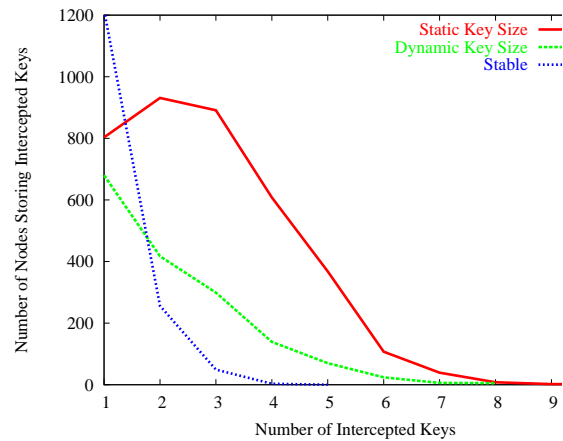
Our experiments were based upon a 2,000 node system organized by the Chord distributed hash table overlay. We used a modified Chord network simulator to accommodate the security requirements of Scribe delineated in Section 3. All tests used the twelve most significant bit (MSBs) of a document as the public identifier, creating  $2^{12}$  (4096) keys system-wide. This gave each node responsibility for two keys. Each node in the experiment internally established two private key stores called valid and stolen. The valid keys were directly associated with the regions the nodes occupied at the current time, and the stolen store contained all keys the node had responsibility for in the past. This scenario assumed an honest-but-curious adversary model, in which all nodes correctly route packets but secretly attempt to decrypt the information. We defined a *compromised key* as any private key stored at two or more different nodes.

We conducted three different types of simulations in which the overlay experienced normal growth, massive growth, and massive failure. We also profiled a stable environment as a baseline where no meaningful growth or failure occurred. At the end of each test we looked at the number of compromised keys and their distribution to determine the security of Scribe in an overlay.

The first set of tests injected all private keys after the bootstrap period (ie. after critical mass in the system was achieved). These tests also assumed that the keys did not expire during the life of simulation. We were surprised by the initial results as the number of compromised keys were quite large. We found that the massive failure operations decreased the number of compromised keys as expected. However, the massive growth simulation produced more compromised keys than anticipated. Figure 2 illustrates the difference in number and distribution of compromised keys, with massive growth producing approximately four times as many duplicate keys as massive failure. Therefore, we recognize join operations as the primary threat to



**Figure 2. Massive growth simulations reveal that join operations lead to 4 times as many compromises as massive failure.**



**Figure 3. In a massive growth period, dynamic key generation yields fewer compromised keys than static keys definition.**

private key security in an overlay.

After analyzing the results, we found a flaw in our initial algorithm. We concluded that injecting all private keys after the system obtained critical mass was not the right solution. Instead, the key escrow system should only generate keys according to the expected size of the network. For example, our initial test injected all 4096 keys (12 MSB as identifiers) into the system immediately after 100 nodes joined the group. This resulted in approximately forty keys being stored at each node. However, the appropriate action would have been to assign the 10 MSB as identifiers, resulting in 1024 keys, or 10 keys per node. As the system grows, so do the number of keys. The key escrow also keeps nodes updated with the number of MSBs currently used. This limits the released keys which in turn limits the number of compromised keys.

Therefore, we ran another simulation in which we dynamically changed the number of MSB for the key identifier. The second algorithm predicted the expected size of the overlay and the key escrow only generated keys to comfortably accommodate the network size. As the network grew, more keys were generated to accommodate the number of nodes.

Our results were as expected. The dynamic algorithm yielded less than half the number of compromised keys. In figure 3, we simulated massive joins and compared the original algorithm against the revised algorithm. Not only were there fewer duplicated keys, but the distribution of the intercepted keys was more localized around the original. By adapting the number of keys to the size of the network, Scribe is able to improve the overall integrity of keys.

Looking again at figure 3, we can see the number of compromised keys is still large. We believe that these compromised keys are localized around the correct lo-

cation in the overlay. However, depending on the security of the message, this threatens its integrity. To address this problem, a slight modification to Scribe would offer high security on selected messages. As a sender encrypts a message to a destination, the node determines the sensitivity of the information. Normal messaging in Scribe will ensure the message gets very close to the target before being decrypted. However, the sending node may determine that only the target should be able to decrypt the message. Therefore, the sender encrypts the message with the full 160-bits of the target. The receiving node then requests a key from the escrow specifically for that 160-bit destination. While this offers increased security, there is a tradeoff in the number of keys a destination node must store.

Our experiments have shown that Scribe is able to efficiently secure communication inside a highly dynamic overlay system.

## 6 Related Work

There are numerous projects involving distributed key systems. The  $\Omega$  project [10] was developed at ATT for key management. This project used threshold cryptography to store signing keys and demonstrated its practicality. However,  $\Omega$  did not support distributed key generation, proactive secret sharing, and was designed for a more trusted environment.

ITTC [13] is a project that explored intrusion tolerance via threshold cryptography. ITTC is fundamentally similar to Scribe in that multiple computers held a private key and collectively performed operations. The project performed evaluation of various threshold schemes and found that performance impact was reasonable. However, the system was designed for small,

trusted, clustered systems, such as Web servers. Scribe looks at large, dynamic machines established in an untrusted network.

Identcrypt [7] is another project being worked on by Boneh et al. The Identcrypt project addressed the rapidly growing problem of email encryption. In this system, email addresses become the public identity key for a person and allow for a more feasible encryption scheme. Scribe is largely based upon this work but is targeted for a very different purpose. Specifically, Scribe addresses a peer-to-peer distributed environment with many untrusting systems.

Secure File System (SFS) [8] decouples the key management and the file system, but their encryption depends on user chosen encryption. Scribe uses encryption for internal communication between nodes, and SFS complements Scribe's design.

Distributed systems such as Condor [1] are exploring key management schemes, but they suffer the same inefficiencies as other distributed systems. Specifically, Condor requires prior knowledge of certificates and a central management component. Additionally, the storage requirements are not scalable and do not compensate for dynamic membership.

## 7 Conclusion

Scribe is a method for efficient key management inside a distributed system that uses identity based encryption. Public resources in a network are addressable by unique identifiers. Using this identifier as a public key, other entities are able to securely access that resource. Scribe provides a key escrow service that authenticates nodes based on their public identity and delivers private keys accordingly. This system allows secure, efficient, authenticated communication inside a distributed system.

Therefore, by using Scribe, there are many advantages to this key management scheme. First, membership is enforced by the key escrow system, which relies on the role a node plays to generate a private key. Next, node identity is certified by the key escrow, and therefore a Certificate Authority is not needed. Each node also does not have to deal with the complexity of key generation or storage required by most systems. Finally, dynamic membership is automatically handled by the system to deliver the correct private keys to a node at a given time.

Our results with Chord show that Scribe is most efficient if it adapts to the changing state of the system. By dynamically generation keys and supporting variable identifier lengths, overlay network messaging becomes more secure. Future work will evaluate Scribe's underlying master key generation and security algorithms.

## References

- [1] Authentication, Authorization, and Privacy in Condor. <http://www.cs.wisc.edu/condor/presentations/PC-2002/hbwang.ppt>.
- [2] D. BONEH, M. FRANKLIN. Efficient generation of shared RSA keys. In *Proceedings Crypto'97*, pp. 425-439.
- [3] D. BONEH, M. FRANKLIN. Identity based encryption from the Weil pairing. In *Proceedings of Crypto '2001, Lecture Notes in Computer Science*, Vol. 2139, Springer-Verlag, pp. 213-229, 2001.
- [4] M. FREEDMAN, E. SIT, J. CATES, R. MORRIS. Introducing Tarzan, a Peer-to-Peer Anonymizing Network Layer. In *International Workshop on Peer-to-Peer Systems*, (2001).
- [5] P. GEMMEL. An introduction to threshold cryptography. In *CryptoBytes*, a technical newsletter of RSA Laboratories, Vol. 2, No. 7, 1997.
- [6] S. HAZEL, B. WILEY. Achord: A Variant of the Chord Look-up Service for Use in Censorship Resistant Peer-to-Peer Publishing Systems. In *International Workshop on Peer-to-Peer Systems*, (2001).
- [7] Identity-Based Encryption Secure Email. <http://crypto.stanford.edu/ibe/>.
- [8] D. MAZIERES, M. KAMINSKY, M. F. KAASHOEK, E. WITCHEL. Separating key management from file system security. In *ACM Symposium on Operating Systems Principles (SOSP'99)* pp 124-139 December 1999.
- [9] S. RATNASAMY, P. FRANCIS, M. HANDLEY, R. KARP, AND S. SHENKER. A Scalable Content-Addressable Network. In *Proceedings of SIGCOMM 2001*, pp 161-172.
- [10] M. REITER, M. FRANKLIN, J. LACY, R. WRIGHT. The  $\Omega$  Key Management Service. In *Journal of Computer Security* 4(4):267-287, (IOS Press, 1996).
- [11] A. SHAMIR. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology - Crypto '84, Lecture Notes in Computer Science*, Vol. 196, Springer-Verlag, pp. 47-53, 1984.
- [12] I. STOICA, R. MORRIS, D. KARGER, M.F. KAASHOEK, AND H. BALAKRISHNAN. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of SIGCOMM 2001*, pp 149-160.
- [13] T. WU, M. MALKIN, D. BONEH. Building Intrusion Tolerant Applications. In *8th USENIX Security Symposium*.