Resilia: a Safe and Secure Distributed Backup System for Small and Medium Enterprises

Christian Damsgaard Jensen, Fernando Meira, and Jacob Nittegaard-Nielsen

Informatics and Mathematical Modeling
Technical University of Denmark
Christian.Jensen@imm.dtu.dk

Abstract. Most small and medium-sized enterprises (SME) operate from a single address, which means that backups are normally kept at the same physical location as the company's computers. This means that fire, flooding or other disasters are likely to destroy both computers and the backups that were meant to ensure the continued operation of the company.
The price per Giga-byte of hard disk storage is falling and at the same time the bandwidth of the connection from small companies to the Internet is increasing, so it appears logical for small companies to achieve improved availability of their backups by storing backups on the hard disk of one or more remote computers. However, storing business-critical information or customer data on a foreign computer requires a mechanism that preserves the secrecy and ensures the integrity of the stored data.
This paper presents Resilia, which is a safe and secure backup system that allows a company to distribute its backup among a number of remote servers, thereby ensuring availability, without compromising the confidentiality and the integrity of the backup. The confidentiality of data in Resilia is ensured with an encryption technique known as threshold cryptography, which means that a backup can be restored even if all cryptographic keys are lost in a disaster. We describe a working prototype of Resilia and report initial performance numbers for the developed prototype.

1 Introduction

The main goals of computer security are normally defined as ensuring the *confidentiality*, *integrity* and *availability* of resources managed by the computer system (these are commonly known as the *CIA properties*). Much of the existing research into computer security has focused on the first two goals, because they are easier to ensure from within the system, but loss of availability, in particular the loss of data stored in business information systems, may have devastating consequences for a company. It is widely reported that companies that experience major data loss face serious problems and may even be forced to close. Examples of such statistics found on the Internet [1] are: "30% of all businesses that have a major fire go out of business within a year.

[1] Many of these quotes are found on the websites of companies that offer backup software or services, but they are normally attributed to an independent source.

Please use the following format when citing this chapter:

Jensen, C. D., Meira, F. and Nittegaard-Neilsen, J., 2007, in IFIP International Federation for Information Processing, Volume 238, Trust Management, eds. Etalle, S., Marsh, S., (Boston: Springer), pp. 383–398.

70% fail within five years"[2], "50% of companies that lose their data in a disaster never open their doors again"[3], "93% of companies that lost their data center for 10 days or more due to a disaster filed for bankruptcy within one year of the disaster. 50% of businesses that found themselves without data management for this same time period filed for bankruptcy immediately"[4] and similar statistics can be found in the press: "Data loss costs U.S. businesses more than $18 billion a year, according to the Pepperdine study. That 2003 study is the most recent estimate available, but Smith says the number is probably much higher today."[1] It is therefore obvious that protecting a company against major data loss is of vital importance to the continued operation of that company.

Frequent and reliable backups have proven to be an important element in the protection against major data loss. By replicating data, it is possible to restore an operational system from the latest backup if the original system is destroyed. Keeping all the replicas in the same location, however, means that whatever happens to one may also happen to the other. Many small and medium enterprises operate from a single address which means that backups are normally kept at the same physical location as the company's computers. This means that fire, flooding or other disasters are likely to destroy both computers and the backups that were meant to ensure the continued operation of the company. Moreover, taking regular backups and managing the (off site) location and rotation of backup media is a cumbersome and repetitive job. While larger companies generally have the resources to accomplish this task, smaller companies generally struggle just to take backups, e.g., "40% of Small and Medium Sized Businesses don't back up their data at all"[5]. Furthermore, the dramatic increase in secondary storage (hard disk space) on most computers, mean that the volume of backups has exploded during the past decade, so backup to magnetic media is not always feasible. The growth in hard disk capacity is matched by a similar growth in bandwidth on the networks that connect small companies to the Internet, so service companies that offer remote storage for network backups have emerged. These service companies solve the problem of ensuring availability of the primary backup data, but they introduce a new problem of protecting the cryptographic keys used to protect the confidentiality and integrity of the backup data. Moreover, these services are charged at a price that exceeds the cost of the hardware needed to host the service within a few months. Developing a relatively simple and cheap networked backup service that ensures the availability of a company's backups without compromising the confidentiality and integrity of its data would therefore benefit many small and medium enterprises.

It is generally agreed that confidentiality is best ensured by keeping a single well guarded copy of the data, while availability is best ensured by keeping many replicas in different physical locations, so there appears to be an inherent conflict between confidentiality and availability. However, there exist cryptographic techniques known as *threshold cryptography schemes*, where the secret information is split into several sets of data known as *shares*, where each share in itself conveys no information about

[2] Attributed to the journal "Home Office Computing Magazine".

[3] Attributed to "University of Texas for research into Info systems".

[4] Attributed to the "National Archives & Records Administration in Washington".

[5] Found on the web site of the Data Deposit Box [2], attibuted to the journal "Realty Times".

the secret, but a previously defined fraction of the shares are enough to reconstruct the original data. The application of such schemes to a distributed backup system means that individual share can be distributed to different backup nodes, which will learn nothing about the secret. If the backup needs to be restored, shares from the predefined fraction of backup nodes have to be retrieved in order to reconstruct the original backup. This addresses the inherent conflict between confidentiality and availability in distributed backup systems. It is easy to imagine that every member of the local commerce council makes backup space available on their servers to the other members of the commerce council. This backup space can then be used in a peer-to-peer (P2P) backup system in the same way that other digital content is shared across the Internet. Other members of the local commerce council can be assumed to be honest about storing and returning data (i.e., integrity and availability), but competitors are likely to try learning the trade secrets of each other. The encryption performed though threshold cryptography, however, means that backups may even be stored on competitors' computers, because each share conveys no information about the overall content.

Different threshold cryptography schemes have been proposed in the literature. We have developed a prototype backup system for small and medium enterprises, called Resilia, which implements two different types of schemes: *secret sharing* and Rabin's *information dispersal algorithm*. These schemes have different properties and allow us to experiment with different trade-offs between computational power and network/storage requirements. Resilia is implemented in Java [3], using the JXTA [4] platform for P2P applications, which ensures platform independency.

The rest of this paper is organized in the following way. Section 2 examines the requirements for network based backup systems. Section 3 provides an overview of threshold cryptography and the Resilia prototype is described in Section 4. The prototype is evaluated in Section 5, related work is surveyed in Section 6 and our conclusions are presented in Section 7.

2 Secure Distributed Backup Systems

Backing up data and information is probably one of the most important aspects of security on a personal computer and for any computer belonging to a distributed system. Every user wants to assure that his work does not evaporate into thin air if some unexpected event happens.

2.1 Network Storage for Backup

Any distributed backup system consists of a local agent on the machine where the backup is made, a number of remote agents on the machines where the backup is stored and a network infrastructure needed to transport the backup from the node where the backup is made to the nodes where the backup is stored.

The backup agent on the client machine is responsible for creating the backup, locating one or more remote nodes that will store the backup, transfer the backup to those nodes in a reliable way and possibly receiving a receipt from the remote backup

node indicating that the backup was received. If the backup needs to be restored, the client requests the remote backup node to return the backup data to the client, which is then able to restore the data.

The backup agent on the remote backup node is responsible for receiving the backup from the client, committing the backup to persistent storage and possibly issuing a receipt to the client. If the client requests the backup, the remote backup node must retrieve the data from persistent storage and return it to the client. This architecture is illustrated in Figure 1.

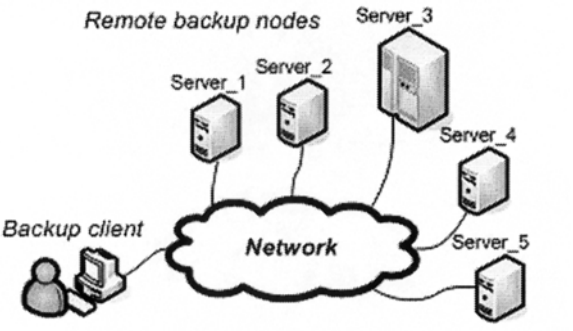

Fig. 1. Backup Network

In order to ensure the availability of the backup, data may be replicated on multiple remote backup nodes. This may either be the task of the backup agent on the client machine or the backup agent on the first remote node that receives a replica of the backup. In the first case, the client negotiates, possibly independent, service level agreements with each remote backup node, while in the second case the remote backup nodes have some predefined agreement between them to store replicas for each other. This agreement is implicit in most cases where the remote backup nodes belong to the same organisation.

2.2 Security Goals

Backups are primarily about ensuring availability and integrity. Having the ability to replace a file by a previous version limits the damage that can be done to the information stored in the file, so the security of the backup system is vital to a company's security strategy. Thus, a *secure backup* can be defined as a backup that has the following proprieties:

Confidentiality: Backups that contain sensitive data should be stored in a way that prevents unauthorised access to the stored data. In particular, backups stored on an insecure site should be protected from unauthorised access. This is normally achieved by encrypting the backup before it is stored on the insecure site, which means that the data received from the insecure site must be decrypted before the backup can be restored.

Integrity: The original data should be returned when a backup is restored. There are several ways that a backup can be modified in a distributed backup system: it may be affected by transmission errors when it is sent over the network, it may be affected by disk errors on the remote backup node or it may be modified by a malicious node that handles the backup data. Integrity is normally achieved by cryptographic means, such as message integrity codes and digital signatures.
Availability: It must always be possible to restore the backup, so the backup system must be able to function despite a fraction of the backup nodes becomes unavailable. The fraction of backup nodes that must be available determines the robustness of the backup system; if this fraction is close to one most nodes must be available and the system is considered *brittle*, but if it is close to zero most nodes may fail and the system is considered *robust*.
Survivability: It must be possible to restore the backup from any machine without special information, such as passwords or cryptographic keys. This ensures that the backup can be restored on a new computer even if all the computers and media belonging to the client's organisation are stolen or destroyed. Survivability is an essential requirement if the backup strategy forms part of a company's business continuity plan.

As mentioned above, confidentiality and integrity are normally achieved by means of cryptography. This introduces the problem of managing the keys needed to encrypt, decrypt or sign data, because these keys cannot be stored locally on the computer. Otherwise, the keys will be lost along with the system in the disaster that caused the need to restore the backup.[6] It is therefore important to develop a secure backup system that does not store cryptographic keys locally on the computer, in order to ensure the survivability of the system.

Although not considered in the definition of a secure backup, *efficiency* is a very important property of any backup-system. Backups are often made when the system is otherwise idle, i.e., during the night. This means that there is a limited time to create the backup and transfer it to the remote backup nodes. Moreover, the system may still be accessed by employees from home or by visitors to the organization's web site, so the backup system cannot utilize the full bandwidth of the organization's Internet connection. The size of the backup should therefore be as small as possible in order to conserve both disk space on the backup nodes and network bandwidth.

2.3 Threat Model

We assume that the client machine has not been compromised; otherwise an attacker would already have access to all the original data in unencrypted form. This means that there are two possible targets for an attacker: the network and the remote backup nodes.

We assume that an attacker may obtain complete control over the network, but that he cannot break strong cryptographic algorithms with brute force (this is commonly known as the Dolev-Yao threat model [5]).

[6] We do not here consider accidental deletion of individual files, which may cause problems, but rarely forces a company into closure.

With respect to the remote backup nodes, we assume that an attacker may have complete control over a limited number of these nodes, e.g., she may have hacked into the node or she may be the legitimate, but untrustworthy, operator of some of the nodes. Hacking systems belonging to different organisations, i.e., where systems configurations and passwords are different, is considerably more difficult than breaking into a set of homogeneous systems belonging to the same organization.

An effective countermeasure against both types of threats to remote backup nodes is therefore to ensure that the remote backup nodes belong to as many different organisations as possible. This also means that backups are likely to disperse over a larger physical area, which improves availability in the case of area specific disasters, such as earthquakes, fires and floods

2.4 Trust Model

We defined four security goals above, where the first three (CIA) concerns the security of the backup while the survivability goal concerns the design and implementation of the backup system itself. If the backup is physically stored by colleagues or competitors, we must assume that they will try to learn any sensitive information stored in the backup, they may also attempt to modify the backup in any way that cannot be detected, but we expect them to return the backup if the client attempts a restore operation. This means that we should not trust the remote backup nodes to maintain confidentiality and integrity, but we do trust that they will actually return backup data when requested, i.e., we trust that help maintain availability.

Consider a small and medium enterprise called ACME, a realistic scenario would be to form a backup network with other companies within the local council of commerce, trade association or professional organisation. Potential competitors are likely to try and obtain trade secrets or customer information from the backup data that they store, so the distributed backup system must protect against this threat. Furthermore, business partners may want to reduce the debt that they owe to ACME or either introduce or increase any debts owed by ACME to the company that stores the backup. The backup system should therefore also have effective integrity protection mechanisms. Finally, the fear of social and commercial ostracism is likely to ensure that they will actually return backup data at ACME's request. In order to sanction backup nodes that do not return backup data when requested and, at the same time, eliminate the risk of framing, the system must the protocol that transfers backup data ensures the non-repudiation of both parties.

3 Threshold Cryptography

A *threshold cryptosystem* requires a number of parties collaborate in the decryption of encrypted data. In the following, we describe two types of threshold cryptosystems, which have the desired property that decryption is possible by combining the encrypted data from the right number of participants, i.e., without requiring a cryptographic key. These types of threshold cryptosystems are a class of algorithms known as *secret sharing* schemes and a single *information dispersal algorithm*.

3.1 Secret Sharing

A *secret sharing scheme* is a model for distributing a *secret* among a group of participants. The secret information is divided into n shares, where n is the number of participants, in such a way that with any m valid shares, where $m \leq n$, it is possible to reconstruct the secret. Any attempt to reconstruct the secret using up to $m - 1$ shares is unfeasible and discloses no information about the secret. This is known as a (m, n)-*threshold scheme.*

A secret sharing scheme is limited in two ways: the size of the shares and random bits. Each share needs to be of at least the same size as the secret itself. If this is not the case, an attacker holding $m - 1$ shares will be able to learn some information about the secret. Assuming that the final share is secret, all $m - 1$ shares still reveal some information. This information cannot be secret, therefore must be random.

Secret sharing was first introduced by Shamir [6] and Blakley [7] independently in 1979. Shamir's secret sharing scheme (SSS) is based on polynomial interpolation, where $n + 1$ points uniquely determine an n-degree polynomial. Knowledge of m shares should suffice to restore the data, so a $(m - 1)$-degree polynomial is generated, which is defined by

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots a_{m-1} x^{m-1}$$

over a finite field. The coefficient $a0$ represents the secret, which is the polynomial intersection with the y-axis. Any other point of the polynomial may be distributed to the n participants. Figure 2 illustrates two examples of polynomials that may be used to implement schemes for $m = 2$ and $m = 3$. The figure shows the secret, point $(0, s)$, and other points that would be distributed to participants of the sharing.

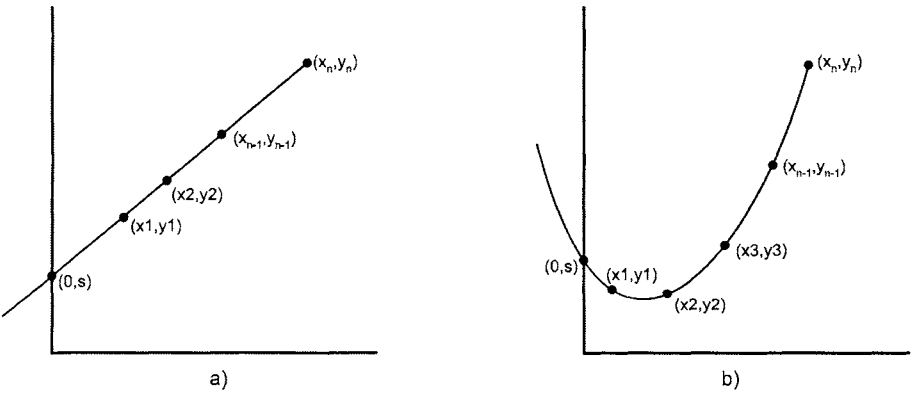

Fig. 2. Polynomials representing: a) $(2, n)$-scheme b) $(3, n)$-scheme

This scheme has two interesting proprieties. It allows different levels of control over the secret, by giving more than one share to selected participants. The more shares a single participant holds, the more control he has over the shared secret, since

fewer participants are needed to reconstruct the secret. The second propriety is that the scheme allows new participants to be added to the sharing after the initial distribution has been made, without affecting the existing shares. Finally, SSS is relatively space efficient; each share only needs to be the same size as the original secret because the x-coordinates of each share can be known to everyone.

Blakley's secret sharing scheme (BSS) is based on the fact that n n-dimensional hyperplanes intersect in a single point. The secret is encoded as a single coordinate of that point, to prevent anyone who knows one or more of the hyperplanes from learning something about the secret. Each participant is given enough information to construct a hyperplane and the secret is recovered by finding the intersection of the planes and selecting the right coordinate.

Both secret sharing schemes have information theoretical security, but BSS requires m times as much storage as SSS and since we have to distribute the shares across the network as part of our backup, we have decided to focus on Shamir's scheme.

Although the secret sharing scheme presented above protects data against passive attacks, like missing shares, there are times when it is necessary to protect against active attacks, such as someone sending incorrect or corrupt shares when distributing shares or even receiving incorrect shares when restoring. This may be accomplished with *verifiable secret sharing* (VSS) schemes. Feldman's VSS scheme [8] is a well-known example that works in the following way: the distribution operation is performed the same way as Shamir's scheme, with the backup client sending each share to each remote backup node in private and also broadcasting a set of values that allows each recipient to verify that their share has not been corrupted in transit. If a user receives a corrupt share a complaint should go back to the backup client. If the backup client receives m or more complaints, then the process fails, otherwise the sender is required to broadcast correct shares to complaining users. Therefore, this scheme ensures a secure system with a cheating sender and at most $m-1$ cheating users, as long as $2(m-1) < n$.

Pedersen also presents a scheme to handle VSS [9]. However, his scheme weakens the protection of integrity of the secret (x) in a trade-off with achieving a true semantic security[7].

A *proactive secret sharing* (PSS) scheme allows a node to generate a new set of shares for the same secret from the old shares without reconstructing the secret. This scheme is very useful for refreshing secret shares in a secure way, such as in server recovery cases. Server break-ins are very hard to detect if the attacker steals some information without modifying anything on the victim host and covering his tracks on his way out. In order to strengthen the security, a PSS scheme may be used periodically. Thus, by refreshing all shares, old shares become useless, so when an attacker tries to steal m shares to recover the secret, these will only be valid during the limited period

[7] A scheme is *semantically secure* if for any function k computable on the secret, the difference $(p_1^{(k)} - p_0^{(k)})$ between the amount of knowledge learned by the adversary by watching the execution of the protocol is negligible, where $p_0^{(k)}$ and $p_1^{(k)}$ are the probabilities that an adversary correctly computes the value $k(x)$ when fed with public information and with additional information gathered during the run of the protocol, respectively.

of time between the start of two PSS operations, i.e., an attacker has to recover all m shares belonging to the same period. With an (m, n) secret sharing scheme, each server holds a share for the entire lifetime of the service, which means that the share is more exposed to threats than a system using PSS.

A safe and secure proactive secret sharing scheme is achieve by combining VSS and PSS schemes, offering a way to protect the secrecy and integrity of sensitive information. One such scheme (the one used in Resilia) was proposed by Herzberg, Jarecki, Krawczyk and Yung [10].

3.2 Information Dispersal Algorithm

The information dispersal algorithm (IDA) was proposed by Rabin [11] in 1987. It is a similar technique to SSS, but it allows a reduction of the communications overhead and the amount of information kept at each site, by sending only partial information to sites. The IDA makes use of a secret key vector, which can be seen as an $n \times m$ matrix, to process the input data. As in the SSS scheme, n stands for the number of participants of the sharing and m the number of required participants to recover the secret. For the main operation in the algorithm, the key matrix is repeatedly multiplied by fix-sized blocks of input data, which can be seen as an $m \times 1$ matrix, until all the data is processed.

We define the key vector as a set of n vectors, $V_1, V_2, \ldots, V_n$, each of length m, $(V_i = (a_{i_1}, \ldots, a_{i_m}))$, with the condition that any subset of m vectors are linearly independent, and the input data as $b_1, b_2, \ldots, b_N$, then the IDA result, c, is achieved by combining values in the following way, for $i = 1, \ldots, n$ and $k = 1, \ldots, N/m$:

$$c_{i_k} = a_{i_1} \cdot b_{(k-1)_{m+1}} + \ldots + a_{i_m} \cdot b_{k_m}$$

The IDA process adds some bits of redundancy that allows some communication errors to be corrected at the reconstruction stage. From each block of a complete key vector computation, that is $c_1, \ldots, c_n$, a value c_i is sent to each participant. It is important to note that there is a link between all n key vectors, all n participants and all n values of each resulting block. Participant number two must always receive value number two of each resulting block, which is represented by vector key number two. This relationship is important at the reconstruction stage.

Thus, each site will store a share of the size $|F|/m$, where $|F|$ is the size of the original data. This represents a total overhead of $((n/m) - 1) \cdot 100$ percent of the size of the original data.

To reconstruct the original data m valid shares are required. With less than m shares, it is infeasible to restore the data. The reconstruction of the data is done by:

$$b_j = a_{i_1} \cdot c_{(1)_k} + \ldots + a_{i_m} \cdot c_{m_k}$$

where $1 \leq j \leq N$. The key vector used for the reconstruction is composed by only the vectors that correspond to the participants' shares used. This results in an $m \times m$ matrix, which is inverted before of the reconstruction of the data.

The advantage of IDA over SSS is that IDA can tolerate the failure of k remote backup nodes by adding $k \cdot |F|/m$ bits to the backup, where SSS requires $k \cdot |F|$ bits to be added.

4 Resilia

The existing prototype of Resilia implements a safe and secure distributed peer-to-peer backup system for small and medium enterprises. Resilia has been designed to satisfy the four security requirements for distributed backup systems (cf. Section 2.2), where traditional cryptography may be used to ensure confidentiality and integrity while threshold cryptography is used to ensure the availability and survivability requirements. The current version of Resilia supports both a combination of a verifiable secret sharing (VSS)[8] scheme and a proactive secret sharing (PSS) scheme [10] and Rabin's information dispersal algorithm (IDA)[9]. The following description of Resilia is very short and we refer to the reports [12, 13] for further details.

Resilia allows any group of users to set up a P2P network, to establish peer groups and to distribute, restore, delete and update their backup files in a secure way. The physical dispersion of the nodes in these peer groups determines the extent of the disaster that the system can tolerate.

A file backup can be performed in two different ways, either using the SSS or the IDA. An overview of the backup agent architecture (both client and remote node) is shown in Figure 3.

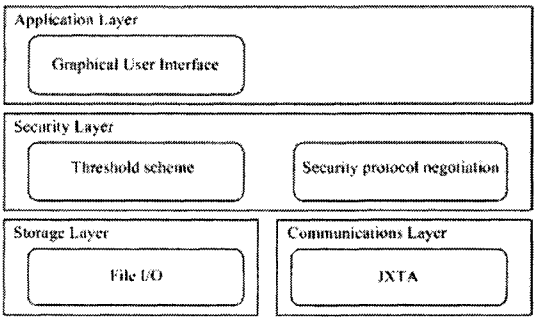

Fig. 3. Resilia Agent Architecture

The Resilia agent architecture is divided into three layers, the *Application Layer*, which manages interaction with the users, the *Security Layer*, which performs the threshold cryptography, authenticates the other nodes in the backup system and establishes secure communication channels with the remote backup nodes.

4.1 Backup and Restore Operations

When a user wishes to backup (part of) her system, she selects the files to be backed up and the threshold scheme to be used in the graphical user interface. The Secu-

[8] The implementation of VSS was developed as part of Jacob Nittegaard-Nielsen's Master Thesis project [12].

[9] The implementation of Rabin's IDA was introduced by Fernando Meira as part of his Final Year project [13].

rity Layer then instantiates the right threshold cryptosystem and starts locating remote backup nodes. The Storage Layer then reads the selected files and creates a backup file, which is handed to the threshold cryptosystem, which performs the necessary threshold cryptography operations [10] and send the shares to the remote backup nodes through the Communication Layer.

If a backup needs to be restored after a disaster, the user selects the restore option from the graphical user interface. The system will then use the lookup service in the Communication Layer to find remote backup nodes that store backups from this node. This requires that the remote backup nodes are able to identify user and authenticate the backup client (cf. Section 4.2). The backup agent will determine which backups can be restored, i.e., where m of n remote backup nodes have responded, and presents them in a list to the user. The user decides which backups to restore (there may be old backups in the list) and starts the restore operation, which performs the necessary threshold cryptography operations to retrieve the backup file. The backup file is then handed to the Storage Layer, which restores the backup on disk.

4.2 Resilia Node Authentication

In order to prevent a malicious backup client from broadcasting a request to restore the backup of another machine, the nodes in the backup system needs an authentication mechanism. Resilia implements an authentication mechanism based on certificates, which contain the identity of the backup agent and the public-key needed in the authentication protocol. It is important to note that we do not require a full public key infrastructure, because certificates may be distributed by other means, e.g., through a mechanism similar to PGP [14] or at certificate exchange sessions at the local council of commerce.

The client certificate prevents malicious backup agents from restoring the backup from another machine, but it also helps ensure availability because, if a client machine is stolen or destroyed, the user may revoke the existing certificate and request a new certificate from the same source where she got the original certificate. This new certificate contains the same ID as the original certificate, which is all that is needed to allow the client to restore the backup.

Servers also have certificates, which prevent a malicious user from flooding the backup network with (virtual) backup servers, in order to receive enough shares to restore the backup of client machines.

5 Evaluation

In the following we present a short analysis of some security properties of the developed prototype, but the main focus will be on a performance evaluation of the two

[10] The operations required to perform threshold cryptography are computation-intensive, but in the case of SSS, we simply encrypt the file using a block cipher (in our case AES) and a randomly generated session-key. This means that we only perform threshold cryptography on a small data structure containing the session-key and some other meta-data describing the backup, so our implementation of SSS is significantly less computation-intensive than IDA.

threshold cryptography schemes, which allows us to compare their relative merit for a backup system designed for small and medium enterprises.

5.1 Security Analysis

Different attacks can be performed to breach the security of both backup clients and servers. An attacker may also attempt to compromise packets send across the network in different ways. Some of these attacks and the countermeasures implemented in Resilia are analysed in the following.

A network packet holds important information, but an attacker has to gather m correct packets in order to compromise a backup file. By "correct" we mean any m of the n packets corresponding to the same backup file. This corresponds to an attacker controlling the right m remote backup nodes for a particular backup. As there are several potential backup nodes for every backup file, the odds that an attacker guesses where to find all shares and breaking in to all those hosts are minimal. If the attacker is able to intercept all packets originating from a backup client, she will know all the shares and will be able to reconstruct the backup, this is why communication between backup agents has to be authenticated and encrypted.

As mentioned before, we expect a fraction of the remote backup nodes to be under the control of one or more attackers. Keeping the number of shares needed to restore the backup relatively high, means that many nodes have to be compromised by the same attacker or different attackers have to collude in order to restore the backup. This does not provide strong security in itself, but it increases the risk of discovery and therefore acts as an effective deterrent against collusion between malicious backup servers.

5.2 Performance Evaluation

The two threshold cryptosystems implemented in Resilia have different requirements with respect to computational power, network bandwidth and storage capacity. We examine these differences in the following.

We have conducted a series of experiments with backup on a local area network, but we have chosen to present the results of a single experiment, where a 5 Kilobyte backup was distributed to 6 nodes where 4 were needed to restore the backup, i.e., we perform threshold cryptography with $n = 6$ and $m = 4$.

The amount of data transmitted over the network and stored on the remote backup nodes is shown in Figure 4. The IDA is expected to distribute $6/4 \cdot 5$ KB, which is effectively what can be seen in Figure 4. The SSS distributes the full encrypted backup and a share of the meta-data to all nodes which means that the total amount of data transmitted is a little more than $6 \cdot 5$ KB, which is also seen in Figure 4. Figure 5 shows the same results, but focus on a different aspect: the data overhead achieved by each scheme. Using the IDA, the defined parameters generate a 50% of added redundancy to the file, which means that total data dispersed over the 6 nodes is equal to the size of the file plus an overhead of 33%. The resulting overhead using the SSS scheme is $|F| \cdot (n - 1)$, which represents more than 80% of the total transmitted data.

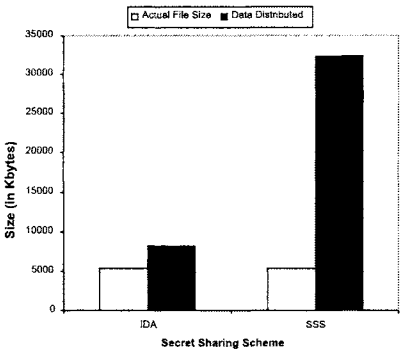

Fig. 4. Volume of backup data distributed to remote nodes.

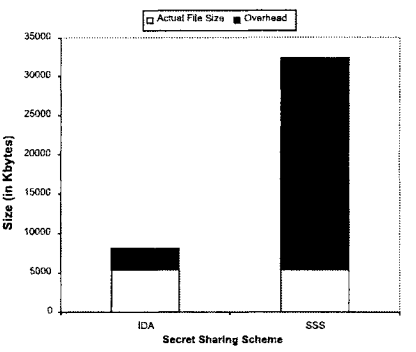

Fig. 5. Communications and storage overhead.

The SSS runs over a small amount of data, the session-key and the meta-data, so the computational power required to construct a share is relatively small. The IDA, on the other hand, runs over the whole file which requires significantly more processing power. We have measured the time needed to perform the necessary threshold cryptography operations on both clients and servers. Figure 6 and 7 show the results of these measurements for backing up and restoring a file. The measured times include all operations on the client that performs the backup and the backup nodes that receive the backup shares and reply with a confirmation message.

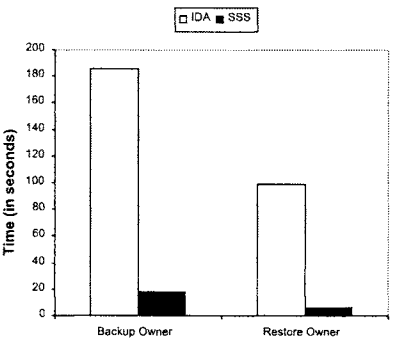

Fig. 6. Computation required on backup client.

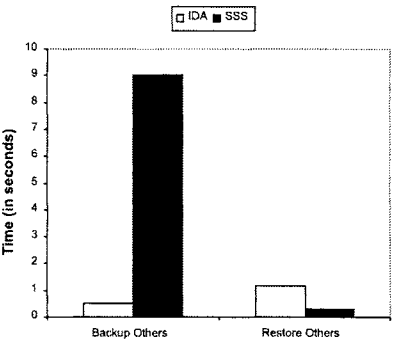

Fig. 7. Computation required on remote backup node

The results, shown in Figure 6, illustrates that if the IDA is used, the backup client has to perform an expensive IDA operation on the entire backup-file for both backup and restore operations, while the SSS only needs to operate on the meta-data which is significantly faster (about 10% of the time required for the IDA). However, the use of a verifiable secret sharing scheme means that some computation is required by the remote backup node when it receives a backup using the SSS, while little processing

is needed by the IDA. The IDA is slightly more expensive when a backup is restored because all nodes have to return a share of the backup file. In the SSS, only one node has to return the encrypted backup file, while all the other nodes only return their share of the meta-data. The measurements shown in Figure 7 do not include the transfer of the encrypted backup file.

5.3 Summary of Evaluation

Our performance evaluation shows that the two threshold cryptography schemes perform very differently when used in a P2P backup system. The SSS based system has a large communications and storage overhead, but requires relatively little computational power. The IDA provides similar availability with significantly less communication and storage overhead, but the IDA has to be applied to the entire backup file, which is a computationally expensive operation.

If the client has limited processing power but a high bandwidth network connection, the SSS should probably be used. On the other hand, plenty of processing power but a network connection with limited bandwidth, then the IDA is probably the best solution.

6 Related Work

A number of secure P2P backup systems have been proposed in the research literature.

pStore combines peer-to-peer systems with techniques for incremental backup systems, which include file encryption and versioning [15]. It shares many of the same goals as Resilia, but off-site storage is fully replicated, requiring higher resource-usage than the IDA, and its security relies on ownership tags. Furthermore, pStore is a pure research project, with no implementation.

DIBS is a freeware backup system that uses Gnu Privacy Control (GPG)[11] to encrypt and sign transactions in order to achieve confidentiality and authenticity [16]. Restore operations require knowledge of the cryptographic keys stored on the local computer, so DIBS does not satisfy the survivability requirement.

Samsara [17] enforces a fair peer-to-peer storage system without requiring trusted third-parties. Peers willing to store data in Samsara have to guarantee that they can provide the same amount of disk space to other peers. It ensures availability and durability through replication, and is used as punishment mechanism for cheating nodes, that have eventually lost data. Samsara was designed as an extension of Pastiche [18].

The CleverSafe Dispersed Storage [19] is an open-source application that is able to disperse a document to 11 storage locations throughout the world. It is implemented in C++ programming language and uses a version of Rabin's IDA to disperse the information. The storage locations are part of a grid, which keeps data private and safe from natural disasters.

[11] GPG is an open source implementation of the well known PGP system.

CleverSafe IDA, also named CleverSafe Turbo IDA, disperses the data into 11 slices, each slice is stored in a different node of the grid. To retrieve the information, at least 6 nodes need to be available. This setup is fixed and cannot be modified. A default grid is already set up and available to users, but users are free to set up their own grid. The advantage of restricting the IDA setup to a 6-out-of-11 scheme for all users is mainly in the ability to optimize the algorithm for this specific case. The optimized algorithm performs significantly better than the general implementation of Rabin's IDA. Although a 6-out-of-11 scheme represents a good balance between availability and storage overhead, it is not possible to shift this balance to suit a particular application. In other words, it is not possible to increase the availability of an important backup, nor reducing the amount of space used to store less important but large backups. Moreover, CleverSafe only supports IDA which we have shown requires significant computational power compared to an implementation of the SSS which only performs threshold cryptography operations on the meta-data.

Another P2P backups system based on IDA is presented by Bella at al. [20]. This work is based on the Chord [21] P2P protocol, which is a proven protocol that supports a dynamic configuration of peers, i.e., new peers may join the network and existing peers may leave the network. The system uses a "meta data file" stored by the backup client, which ensures that only the owner may initiate a restore operation. However, this also means that the backup cannot be restored if the client is stolen or completely destroyed, so their system does not meet the survivability requirement that we defined for Resilia. Finally, no evaluation of the implementation has been reported, so it is unclear to what degree the system actually works.

7 Conclusions

In this paper we addressed the problem of maintaining safe and secure off-site backups in small and medium enterprises. We proposed to build a safe and secure backup service based on threshold cryptography, and presented Resilia which is our prototype implementation of this proposal.

Resilia implements two types of threshold cryptography, secret sharing and Rabin's information dispersal algorithm. We showed that combining efficient block ciphers with secret sharing of keys and meta-data produces a system with low computational overhead, but full copies of all backup data have to be distributed to and stored on all the remote backup nodes. The implementation of Rabin's IDA requires significantly less communication and remote storage, but requires significantly more computational power. The best choice of threshold cryptography therefore depends on the backup client's environment.

Although Resilia was developed for small and medium enterprises, we believe that the techniques presented in this paper may also benefit large providers of backup services, who would only need to operate a single facility and rely on mutual arrangements with competitors to ensure replication of their clients' data.

References

1. S. Armour (2006) Lost digital data cost businesses billions. In USA TODAY 12 June 2006.
2. Data Deposit Box (2007) Data Loss Quotes and Statistics. Available at URL: `http://www.datadepositbox.com/media/data-loss-statistics.asp`, visited 21 February 2007.
3. Sun Developer Network (2007) Java Tecchnology: Reference. Available at URL: `http://java.sun.com/reference/index.html`, visited 21 February 2007
4. J. D. Gradecki (2002) Mastering JXTA: building Java peer-to-peer applications. Wiley Publishing
5. D. Dolev, A. C. Yao (1981) On the security of public key protocols. In Proceedings of the IEEE 22nd Annual Symposium on Foundations of Computer Science, pp. 350–357
6. A. Shamir (1979) How to share a secret. In Communications of the ACM, vol. 22, no. 11, pp. 612–613
7. G. R. Blakley (1979) Safeguarding cryptographic keys. In AFIPS 1979 NCC, Vol. 48, pp. 313–317.
8. P. Feldman (1987) A practical scheme for non-interactive verifiable secret sharing. In Proceedings of the 28th IEEE Symposium on Foundations of Computer Science (FOCS '87)
9. T. P. Pedersen (1991) Non-interactive and information-theoretic secure verifiable secret sharing. In Proceedings of Crypto'91 (LNCS 576), pp. 129–140
10. A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung (1995) Proactive secret sharing or: How to cope with perpetual leakage. In Proceedings of Crypto'95 (LNCS 963)
11. M. O. Rabin (1989) Efficient dispersal of information for security, load balancing, and fault tolerance. In Journal of the ACM, Vol. 36, No. 2, pp. 335–348
12. J. Nittegaard-Nielsen (2004) Sikkert og plideligt peer-to-peer filsystem. Master's thesis, Technical University of Denmark (in Danish)
13. F. Meira (2005) Resilia: A safe & secure backup-system. Final year project, Engineerng Faculty of the University of Porto
14. S. Garfinkel (1994) PGP: Pretty Good Privacy. O'Reilly
15. C. Batten, K. Barr, A. Saraf, and S. Trepetin (2002) pStore: A secure peer-to-peer backup system. Technical Memo MIT-LCS-TM-632, Massachusetts Institute of Technology Laboratory for Computer Science
16. E. Martinian (2007) Distributed internet backup system (dibs). Available at URL `http://www.csua.berkeley.edu/~emin/source_code/dibs`
17. L. P. Cox and B. D. Noble (2003) Samsara: honor among thieves in peer-to-peer storage. In Proceedings of the nineteenth ACM symposium on Operating systems principles
18. L. P. Cox, C. D. Murray, and B. D. Noble (2002) Pastiche: making backup cheap and easy. SIGOPS Opererating Systems Review, 36(SI):285–298
19. CleverSafe Project (2007) Cleversafe dispersed storage project. Available at URL `http://www.cleversafe.org/wiki/Cleversafe_Dispersed_Storage`
20. G. Bella, C. Pistagna, S. Riccobene (2006) Distributed Backup through Information Dispersal. In Electronic Notes in Theoretical Computer Science, Vol 142:63–77
21. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan (2001) Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In Proceedings of the 2001 ACM SIGCOMM Conference

All websites were last visited in February 2007.