Mobile Proactive Secret Sharing in Cloud Computing

*Vaibhav Kumar, **R. P. Ojha

*M. Tech. Scholar, Mewar University, Chittorgarh, Rajasthan
**Associate Professor, GCET, Gr. Noida, India

ABSTRACT

This research paper introduces a scheme to secure any secret value in cloud network by Mobile Proactive Secret Sharing (MPSS). This is an extension of proactive secret sharing, where contributing parties of a network hold the shares of a secret value. Mobile proactive secret sharing is much more flexible than proactive secret sharing in terms of group membership: instead of the group of shareholders being exactly the same from one instance to the next, we allow the group to change arbitrarily. In addition, we allow for an increase or decrease of the threshold at each instance.

Key Words: Cloud Computing, Proactive secret sharing, cryptography, epoch, Byzantine faults.

1. INTRODUCTION

Secret sharing allows a collection of parties to possess shares of a secret value (such as a secret key), such that any t + 1 shares can be used to reconstruct the secret, yet any t shares provide no information about the secret. Sharing of cryptographic keys is crucial in cloud network intended to withstand Byzantine faults, that is, failures that cause servers to behave arbitrarily badly, perhaps because they have been compromised. This is in contrast to systems that tolerate only fail-stop failures, in which servers stop responding altogether. In the context of Byzantine faults, secret sharing allows systems to perform cryptographic operations securely, preserving the secrecy of the keys despite up to t malicious servers.
In long-lived systems, however, servers can be compromised over time, giving an adversary the opportunity to collect more than t shares and recover the secret. Additionally, systems may fail to function properly, for instance due to hardware failure or an attack. To prevent the number of failures from exceeding the threshold the system is designed to tolerate, servers must be repaired or replaced over time, perhaps with newly-installed servers. Moreover, this replacement must be performed periodically even in the absence of detected faults due to the potential for lie-in-wait attacks. In this type of attack, faulty servers appear to behave correctly while an attacker compromises additional machines; once t+1 servers have been compromised, they start behaving badly or simply reveal the secret to the attacker.

Proactive secret sharing (PSS) schemes, address the problem that shares can be exposed or lost over time due to Byzantine faults. In PSS, servers execute a share regeneration protocol, in which a new set of shares of the same secret is generated and the old shares discarded, rendering useless any collection of t or fewer shares the adversary may have learned. Furthermore, PSS schemes typically provide a share recovery protocol so that a full set of new shares can be generated even if some of the old shares (up to some maximum number of faults tolerated) have been lost.

As we know that there are many security related issues are in cloud computing, this security scheme can be applied in cloud computing to secure secret values or data.

2. SECRET SHARING

Secret sharing was first proposed by Shamir and independently by Blakley. These seminal schemes operate under a very simple model: a trusted dealer has a secret and distributes a different share of that secret to each server. Shamir demonstrates that a passive adversary who learns up to t shares of the secret gains no partial information about the secret, yet any t + 1 servers can combine their shares to recover the secret. Blakley's scheme makes a similar guarantee, except that it does not provide perfect secrecy; combinations of t or fewer shares reveal partial information about the secret, and additional modifications are needed to ensure perfect secrecy. Shamir's scheme is based on interpolation of polynomials over a finite field, whereas Blakley's scheme encodes the secret as an intersection of n-dimensional hyper planes. Each share in Shamir's scheme is the same size as the original secret, but shares in Blakley's scheme are t times as large. Shamir's scheme is more widely used because it provides stronger guarantees and better space efficiency using only relatively simple mathematics. The PSS scheme of Zhou et al. is based on a different secret sharing mechanism that is even simpler but more limited than both Shamir's and Blakley's schemes.

3. VERIFIABLE SECRET SHARING

Feldman [Fel87] and Pedersen [Ped91a] introduced verifiable secret sharing schemes based on Shamir's work. These schemes allow shareholders to determine whether the dealer sent them valid shares of the secret, hence allowing them to come to a consensus regarding whether the secret was shared successfully. In this context, the dealer is semi-trusted; it does not reveal the secret, but it might attempt to fool servers into accepting an invalid sharing of the secret. Verifiable secret sharing is an important component in many distributed secret sharing protocols involving untrusted participants because the protocols typically involve each server acting as a semi-trusted dealer to all of the others.

Feldman's and Pedersen's schemes have similar efficiency but slightly different security guarantees. Feldman's scheme is perfectly binding (meaning that an untrusted dealer cannot fool shareholders into accepting an invalid sharing) and computation- ally binding (meaning that secrecy is subject to computational hardness assumptions and the amount of computation available to the shareholders). Pedersen's scheme, on the other hand, is computationally binding and perfectly hiding. It has been shown that schemes that are both perfectly hiding and perfectly binding do not exist. In the context of the proactive secret sharing schemes we discuss, a computationally unbounded attacker can exploit the VSS to expose the secret regardless of which choice we make, so the distinction is a non-issue for us.

4. PROACTICE SECRET SHARING

4.1 Ostrovsky and Yung

Proactive secret sharing was introduced by Ostrovsky and Yung in [OY91] as a way to cope with network worms or viruses. In their model, an adversary infects shareholders at a constant rate, but shareholders are also rebooted and restored to their correct state at an equal rate. Hence, they assume that in any given time period (we use the term *epoch* herein), $t < \lfloor n/2 \rfloor$ shareholders may be faulty. (Note that this threshold is better than the $t < \lfloor n/3 \rfloor$ typically required for asynchronous schemes, and is possible only because the correctness of their protocol is based on the unrealistic synchrony assumption that servers that fail to respond within some fixed amount of time are faulty.) Shareholders preserve the privacy of the shared secret by executing a *refresh protocol* to generate a new sharing of the secret, discarding their old shares, and using the new shares for the next epoch. In [OY91], the refresh protocol is implemented via a generic secure multi-party computation protocol on the existing shares. These multi-party protocols (e.g., [BGW88, CCD88, RBO89]) are general but inefficient. Some are implemented in terms of many instances of verifiable secret sharing, with the number of rounds proportional to the depth of a circuit that implements the function to be computed.

This seminal work is important because it was the first to demonstrate that proactive secret sharing is theoretically possible; however, the Ostrovsky and Yung scheme is infeasible in practice because performing nontrivial calculations using generic se-cure multi-party protocols is expensive. Furthermore, Ostrovsky and Yung assume that the network is synchronous, and that secure channels are uncompromised by past corruptions of the endpoints. Practical implementations of secure channels involve secret keys that would be exposed by a compromise of the endpoints, and hence it is unclear how to recover the node in that case, since the adversary now knows the node's secret keys. Also, although they show that "recovery" of a machine's state is possible in theory by having all of the other participants construct it via a secure multi-party computation, it is unclear how one might perform recovery efficiently in practice.

4.2. Herzberg et al

Herzberg et al. [HJKY95] address the efficiency problem by introducing a protocol specialized to the problem of generating a new secret sharing. In their scheme, participants numbered $i = 1 \ldots n$ have an initial Shamir sharing with polynomial P (i.e., secret $s = P(0)$ with shares $P(1) \ldots P(n)$), and in the refresh protocol they construct a new sharing $P + Q$, where Q is a random polynomial with $Q(0) = 0$. To handle the case where a previously-corrupted node k has lost its old share and needs to recover its correct state, other participants execute a *recovery protocol* in which each other party i sends $P(i) + R_k(i)$ to k, where R_k is a random polynomial with $R_k(k) = 0$. Note that in an asynchronous network, recovery may additionally be needed for nodes that have never been faulty, simply because they never received their share from a previous execution of the protocol.

Herzberg et al.'s scheme is difficult to translate into an asynchronous network protocol partly because it has an accusation/defense phase in which the network is assumed to be reliable. Each server sends a message to each other server, and if any senders misbehave, the recipients broadcast accusations against them. Then the accused servers must broadcast a defense, or else they will be deemed faulty by the other servers. However, in an asynchronous network, we do not know how long it will take for us to receive defenses from honest servers, and if we establish a specific timeout, we may spuriously deem honest servers to be faulty if their responses are delayed. The authors claim in a footnote that for certain encryption schemes such as RSA, the defense step can be eliminated, which might simplify the translation. How- ever, we show that the encryption scheme must also be forward-secure. Furthermore, fixing the problem in an asynchronous network requires a property of the encryption primitive that is stronger than chosen cipher text security, and neither RSA nor RSA under the Fujisaki-Okamoto transformation [FO99] satisfy this property. (More specifically, in addition to revealing the plaintext, the decryption oracle discloses all randomness used in the encryption computation.)

4.3 Cachin et al.'s Asynchronous Scheme

The protocol of Cachin, Kursawe, Lysyanskaya, and Strobl [CKLS02] is the first efficient scheme in the asynchronous model, also for $t<\lfloor n/3 \rfloor$. Whereas the Herzberg et al. scheme [HJKY95] computes each new share $P'(i)$ as a function of a *corresponding* old share $P(i)$, the Cachin scheme is based on *resharing* the shares of the secret and combining the resulting sub shares to form new shares of the secret. Their paper first presents a protocol for asynchronous verifiable secret sharing, then shows how to build an asynchronous proactive secret sharing scheme by having each honest shareholder create a VSS of its share. Their VSS scheme is similar to the one of Stinson and Wei [SW99], and the method of computing new shares from sub shares is based on the linearity of Lagrange interpolation, which was proposed by Desmedt and Jajodia in [DJ97]; however, the authors seem to be unaware of either of these earlier works.

To handle share recovery for participants who have lost or never received their shares, Cachin et al. use a two-dimensional sharing $P(\bullet, \bullet)$ in which shares are one- dimensional projections $P(i, y)$ and $P(x, i)$; thus, any participant can interpolate its share given the points of overlap with at least $t + 1$ other shares. Cachin et al.'s protocol requires that a significant amount of information be broadcast by each par- ticipant to each other participant even in the absence of faults, whereas our scheme achieves better efficiency in the common case by using a coordinator. Moreover, their protocol does not support changing the set of shareholders.

5. MOBILE PROACTIVE SECRET SHARING

5.1 The Desmedt and Jajodia Scheme

Desmedt and Jajodia [DJ97] were the first to propose an extension of proactive secret sharing, which they call secret redistribution and we call mobile proactive secret sharing that allows the set of shareholders, number of shareholders, and threshold to change. They use the same strategy as Cachin et al. [CKLS02] (albeit in more generic group-theoretic terms), in which each "old" shareholder acts as a dealer and shares its share of the secret to the new shareholders. The new shareholders then combine the sub shares from some set of at least $t + 1$ old shareholders to produce new shares of the secret. However, their scheme is not verifiable, and thus faulty nodes in the old group that behave incorrectly can cause the new shareholders to generate an invalid sharing of the secret. Furthermore, their scheme is not formulated in terms of a concrete network protocol, so it is unclear, for instance, how the new shareholders are to decide which old shareholders to accept shares from if there are faulty old shareholders and lost network messages. A direct implementation of their proposal would only work in a synchronous network with a passive adversary that can eavesdrop and corrupt nodes, but not generate spurious messages.

5.2 Wong, Wang, and Wing Scheme

Wong, Wang, and Wing [WWW02] improve upon Desmedt and Jajodia [DJ97] in two significant ways. First, they provide a complete, implementable, network protocol. Second, their scheme is verifiable, so cheating "old" shareholders can't compromise the validity of the sharing or prevent it from completing. However, their scheme relies upon all of the new shareholders being honest for the duration of the protocol, which is an unrealistic assumption. Furthermore, their scheme is inefficient in the presence of malicious old shareholders because it gives the new shareholders no way to determine which old shareholders sent bad information. Hence, they must restart their protocol potentially an exponential number of times using different subsets of old shareholders, until a set of entirely honest shareholders is chosen.

5.3 APSS

Zhou et al. [ZSvR05] proposed the first technique that works in an asynchronous model, which they call APSS. Here the threshold is modified from $t <\lfloor n/2\rfloor$ to $t < \lfloor n/3\rfloor$, which is optimal for protocols relying upon asynchronous Byzantine agreement [CL02].

Their construction is based on an exclusive-or sharing scheme rather than on Shamir's secret sharing. The exclusive-or scheme is simpler and more limited because it only supports k-out-of-k sharings, i.e., all the shares are required to reconstruct. In the exclusive-or scheme, given a secret s, generate random values $r_1, r_2, \ldots, r_{t-1}$ and output shares $r_1, r_2, \ldots, r_{t-1}, r_1 \oplus r_2 \oplus \bullet\bullet\bullet r_{t-1} \oplus s$, where $\oplus$ denotes bitwise exclusive or.

Any combination of $k - 1$ of these shares is indistinguishable from random values, but the exclusive-or of all of the shares is s. For every possible subset of honest shareholders of size $t + 1$, they produce a trivial $t + 1$-out-of-$t + 1$ sharing of the secret using the exclusive-or sharing; hence, any $t+1$ shareholders can reconstruct the secret. However, this construction results in exponentially large shares of the secret; hence, the communication required to refresh those shares is exponential in n, the number of shareholders. Chen [Che04] implemented and analyzed their scheme and found the communication overhead (total data exchanged for all servers) to be 47 kB for $t = 1$, 3.4 MB for $t = 2$, 220 *MB* for $t = 3$, and unacceptable for larger thresholds, at least in her implementation. Unfortunately, it seems that in order to ensure that the probability that the threshold is exceeded is reasonably small in a real-world system, using realistic assumptions about failure rates, the value of t must be 6 or greater [Rod]. Hence, to be practical, it seems the protocol must have sub exponential complexity, regardless of optimizations we might be able to apply to this exponential scheme. Our protocol requires $O(n^4)$ bytes of network traffic with reasonable constant factors.

5.4 MPSS — Our Scheme

Our approach uses a simple Feldman VSS, and the technique for generating new shares is based on the one of Herzberg et al [HJKY95]. However, our protocol assumes a much weaker (asynchronous) network and allows the group to change. When the group changes, it is able to handle a threshold of up to t Byzantine shareholders in the old group and an additional threshold of t Byzantine servers in the new group. Furthermore, unlike the scheme of [WWW02], we achieve worst-case polynomial communication complexity, and moreover our protocol has low overhead in the optimistic case where there are no failures.

Our protocol makes use of accusations as part of choosing the new shares, as does Herzberg et al. However Herzberg et al. make use of an accusations/defense phase, which require extra interaction that is undesirable in the asynchronous setting. In particular, when servers receive invalid messages, they must accuse the sender, and if the sender is honest it must broad- cast a defense to prove that the accusation is specious. But if message delays can be arbitrary, it is impossible to ensure that all accusations and all defenses from honest parties have been received, and hence we cannot tell which servers are misbehaving. Our protocol does not require accusations, but as an optimization, this thesis presents an optional extension called *verifiable accusations*. Unlike Herzberg et al.'s accusations, verifiable accusations require no defense phase, as any party can determine the validity of the accusation.

6. CLOUD COMPUTING

Cloud computing is the delivery of computing and storage capacity as a service to a heterogeneous community of end-recipients. The name comes from the use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts services with a user's data, software and computation over a network.

7. SECURITY ISSUES IN CLOUD COMPUTING

As cloud computing is achieving increased popularity, concerns are being voiced about the security issues introduced through adoption of this new model. The effectiveness and efficiency of traditional protection mechanisms are being reconsidered as the characteristics of this innovative deployment model can differ widely from those of traditional architectures. An alternative perspective on the topic of cloud security is that this is but another, although quite broad, case of "applied security" and that similar security principles that apply in shared multi-user mainframe security models apply with cloud security.

The relative security of cloud computing services is a contentious issue that may be delaying its adoption. Physical control of the Private Cloud equipment is more secure than having the equipment off site and under someone else’s control. Physical control and the ability to visually inspect the data links and access ports is required in order to ensure data links are not compromised. Issues barring the adoption of cloud computing are due in large part to the private and public sectors' unease surrounding the external management of security-based services. It is the very nature of cloud computing-based services, private or public, that promote external management of provided services.

This delivers great incentive to cloud computing service providers to prioritize building and maintaining strong management of secure services. Security issues have been categorized into sensitive data access, data segregation, privacy, bug exploitation, recovery, accountability, malicious insiders, management console security, account control, and multi-tenancy issues. Solutions to various cloud security issues vary, from cryptography, particularly public key infrastructure (PKI), to use of multiple cloud providers, standardization of APIs, and improving virtual machine support and legal support.

Cloud computing offers many benefits, but it also is vulnerable to threats. As the uses of cloud computing increase, it is highly likely that more criminals will try to find new ways to exploit vulnerabilities in the system. There are many underlying challenges and risks in cloud computing that increase the threat of data being compromised. To help mitigate the threat, cloud computing stakeholders should invest heavily in risk assessment to ensure that the system encrypts to protect data; establishes trusted foundation to secure the platform and infrastructure; and builds higher assurance into auditing to strengthen compliance. Security concerns must be addressed in order to establish trust in cloud computing technology.

8. MPSS IN CLOUD COMPUTING TO AVOID SECURITY ISSUES

Mobile proactive secret sharing scheme can be used in cloud network to secure data and other secret values. As in cloud computing, many independent computing systems are connected together for a particular job, then the important information of this job can be subdivided into thresholds for individual computing systems. These computing system then store these thresholds.

If an attacker tries to access that information in any individual system, finally he can get only some encrypted or coded part of that information. In cloud computing network system, it is not easy to enter or unauthorized access of every system because every system may have different functionalities like operating system, firewall system, software etc.

In comparison to other computing system networks, where operating systems are almost same on all systems, MPSS scheme work better in cloud network.

9. CONCLUSION AND FUTURE WORK

By this research paper, we found that security in cloud computing is a big issue. Taking the benefits of the different functionalities of cloud computing systems in a cloud network where they have different operating systems and different other system software and application software, we can apply mobile proactive secret sharing scheme to avoid security attacks. When important information is distributed among systems of cloud network and stored in some encrypted form, it is not easy for a attacker to access all systems of the network to get the information.

This research paper may lead to overcome the security issues of a cloud computing network. As systems of cloud computing having different operating systems and different security related settings are connected together on some job, MPSS will be very useful in avoiding the security attacks.

REFERENCES

[1]D. Boneh, X. Boyen, and S. Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In *RSA Conference*, pages 226-243, 2006.

[2]D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology—CRYPTO 2001*, Lecture Notes in Computer Science, pages 213-229. Springer-Verlag, 19- 23 August 2001.

[3]Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the Twentieth Annual ACM Sym- posium on Theory of Computing*, pages 1-10, Chicago, Illinois, May 1988.

[4]G.R. Blakley. Safeguarding cryptographic keys. In *Proc. AFIPS 1979*, volume 48, pages 313-317, June 1979.

[5]D. Bleichenbacher. Chosen ciphertext attacks against protocols based on RSA encryption standard PKCS #1. pages 1-12, 1998.

[6]Mihir Bellare and Sara Miner. A forward-secure digital signature scheme. In Michael Wiener, editor, *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431-448. Springer-Verlag, 15-19 August 1999. Revised version is available from http://www.cs.ucsd.edu/ mihir/.

[7]Ran Canetti and Shafi Goldwasser. An efcient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In *Theory and Application of Cryptographic Techniques*, pages 90-106, 1999.

[8]Kathryn Chen. Authentication in a reconfigurable byzantine fault tol- erant system. In *MEng Thesis, Massachusetts Institute of Technology*, 2004.

[9]R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key en- cryption scheme. In Eli Biham, editor, *Advances in Cryptology—EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255-271. Springer-Verlag, 4 - 8 May 2003.

[10]M. Castro and B. Liskov. A Correctness Proof for a Practical Byzantine- Fault-Tolerant Replication Algorithm. Technical Memo MIT/LCS/TM- 590, MIT Laboratory for Computer Science, 1999.

[11]Miguel Castro and Barbara Liskov. Practical Byzantine Fault Toler- ance and Proactive Recovery. *ACM Transactions on Computer Systems*, 20(4):398-461, November 2002.

[12]Don Coppersmith, editor. *Advances in Cryptology—CRYPTO '95*, vol- ume 963 of *Lecture Notes in Computer Science*. Springer-Verlag, 27-31 August 1995.

[13]R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proceedings of the 10th (ACM) Symposium on the Principles of Dis- tributed Computing*, pages 51-61, 1991.

[14]Torben Pryds Pedersen. Non-interactive and information-theoretic se- cure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129-140. Springer-Verlag, 1992, 11-15 August 1991.

[15]Torben Pryds Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In D. W. Davies, editor, *Advances in Cryptology—EUROCRYPT 91*, volume 547 of *Lecture Notes in Computer Science*, pages 522-526. Springer-Verlag, 8-11 April 1991.

[16]S. Pohlig and M. Hellman. An improved algorithm for computing log- arithms over GF(p). *IEEE Transactions on Information Theory*, IT-24:106-110, 1978.

[17]T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty pro- tocols with honest majority. In *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 73-85, New York, NY, USA, 1989. ACM Press.

[18]Rodrigo Rodrigues et al. Automatic reconfiguration for large-scale dis- tributed storage systems. Unpublished.

[19]David Andrew Schultz, Mobile Proactive Secret Sharing, Massachuates Institute of Technology, USA, 2007