Threshold Cryptosystems in Asynchronous Networks

Aleksandra Sešić[1], Veljko Malbaša[2]

Abstract–**Threshold cryptography makes it possible to design cryptographic systems in which some operations require the collaboration of several users. As a result, security of the system is increased. This paper aims at guiding readers into this interesting field under the asynchronous message-passing model of distributed computing.**

Keywords–**Threshold cryptosystem, verifiable secret sharing, proactive security, asynchronous system model.**

I. INTRODUCTION

Distributed cryptography, introduced in 1987, is a variant of traditional cryptography intended for distributed services. Several distributed cryptosystems have been proposed until now. Most of them have a threshold structure, which means that both the sets of corrupted servers that must be tolerated by the system and the sets that are qualified to perform some action are determined by their cardinality. Due to this fact, distributed cryptography is called also in general *threshold cryptography*. The surveys of threshold cryptography can be found in [1,2].

Threshold cryptosystem is a public key cryptosystem in which the secret key is shared among the set of users. Only some qualified subsets of users will be able to perform the operation related to the secret key (decrypting or signing). In this way, security of the system is increased, because the loss or theft of several shares of the secret key does not necessarily break the system's security.

We are especially interested in protocols that require no interaction or synchronization among the servers, and as such can be efficiently run on an *asynchronous communications network* where messages are delivered with arbitrary delay and in which the speeds of the nodes can get out of synchronism to an arbitrary extent. It is the weakest model (i.e., methods for this model work in other models, too) and the most realistic for distributed computing in today's large scale wide-area networks such as the Internet.

II. THRESHOLD CRYPTOSYSTEMS

A *threshold public-key cryptosystem* [3] looks similar to an ordinary public-key cryptosystem with distributed decryption. Given a ciphertext resulting from encrypting some message and more than $t+1$ valid decryption shares for that ciphertext, in a system which tolerates up to t faults, it is easy for a client to recover the message; this property is called *robustness*.

This means that corrupted players should not be able to prevent uncorrupted servers from decrypting ciphertexts, i.e., that the decryption service is available even if the adversary can send bad decryption shares. The key to robustness is validity checking, based on a public key system, where one can ignore all incorrect decryption shares without exhaustive searching to find out who sent the wrong decryption share. Notice that the message can be recovered *without revealing* the secret decryption key.

The scheme must also be secure against *chosen ciphertext attacks* [4] in order to be useful for all conceivable applications. This type of attack is one in which a cryptanalyst attempts to determine the *key* from knowledge of plaintext that corresponds to ciphertext selected (i.e., chosen) by the analyst. This type of attack is generally most applicable to public-key cryptosystems, for once the (private) key is known, all subsequent messages from the same source can be deciphered. For the threshold case, security means that the adversary cannot obtain any meaningful information from a ciphertext unless she has obtained a corresponding decryption share from at least one honest party.

In a *threshold signature scheme*, each server holds a share of the secret signing key and may generate shares of signatures on individual messages upon request. A *signing* algorithm takes as inputs a message, the public key and the secret key share. It outputs a signature share on the submitted message. The validity of a signature share can be verified for each server by a *share verification* algorithm. It takes as inputs a message, a signature share on that message from a server S_i along with the public key and local verification key of S_i. A *share combining* algorithm takes as input a message and $t+1$ valid signature shares on that message, along with public keys and the verification keys and outputs a valid digital signature on that message *without knowing* the actual secret signing key. This key speaks for the service but is never materialized at individual servers comprising the service. The signature can later be verified using the single, publicly known signature verification key. Notice that in particular, the threshold approach rules out the naïve solution based on traditional secret sharing, where the secret key is shared in a group but reconstructed by a single player each time when a signature is to be produced. Such a protocol would contradict the requirement that no t (or less) players can ever produce a new valid signature.

The two basic security requirements are *non-forgeability* and *robustness*.

Non-forgeability property means that t or less corrupted servers will not be able to forge signatures, i.e., to provide a valid signature on a message for which no honest party generated a signature share.

Robust threshold signature scheme can withstand the participation of dishonest signers during the signature computation operation. This is a mechanism that succeeds in constructing a valid signature even if the partial signatures

[1] Aleksandra Sešić is with Nopal, 21400 Bačka Palanka, Serbia and Montenegro, Email: asesic@nopal.co.yu
[2] Veljko Malbaša is with the School of Engineering, University of Novi Sad, 21000 Novi Sad, Serbia and Montenegro, Email: malbasa@uns.ns.ac.yu

contributed by some of the signers are incorrect. Due to robustness, corrupted servers will not be able to prevent the uncorrupted servers from computing correct signatures, i.e., it is infeasible for a computational bounded adversary to produce *t+1* valid signature shares that cannot be combined to a valid signature.

Threshold-cryptographic protocols ensure security as long as at most *t* of servers are broken into. They enhance the security against break-in attacks in many scenarios. However, threshold cryptography is also limited. Given sufficient amount of time, an attacker can break into servers one by one, thus eventually compromise the security of the system. This danger is particularly eminent in systems that must remain secure for long periods of time (such as certification authorities) or where secure recovery may be difficult (such as with secure communication).

Proactive security is a mechanism for protecting against such long-term attacks. Proactive cryptosystems operate in phases. They can tolerate the corruption of up to *t* different servers in every phase [5]. That is, first distribute the cryptographic capabilities among several servers. Next, have the servers periodically engage in a *refreshment protocol* that proactively reboots all servers at the beginning of every phase and subsequently refreshes the secret key shares. Knowledge of the shares from the previous phases becomes useless to attack the system in the future. This protocol will allow the servers to automatically recover from possible, undetected break-ins, and in particular will provide the servers with new shares of the sensitive data while keeping the sensitive data unmodified.

Share refreshing is a distributed protocol and in all proactive cryptosystems it relies on verifiable secret sharing. Verifiable secret sharing is a fundamental primitive in distributed cryptography [6] that has found its application in threshold cryptosystems. A verifiable secret sharing protocol allows each shareholder to verify that the share is consistent with other shares in case the dealer of shares might be faulty.

III. Cryptography versus Distributed Computing

The field of multi-party cryptographic protocols is where cryptography and distributing computing meet [7]. However, this field is considered as a part of cryptography, which is the consequence of the dominant role of cryptographic notions and techniques in the current research of cryptographic protocols. Most of the cryptographic research is concerned with two-party computations where typically an asynchronous message passing model is assumed (almost always implicitly). For multi-party cryptographic protocols a synchronous model consisting of either point-to-point channels or a single broadcast channel is used most frequently. Results for asynchronous communication and arbitrary networks of point-to-point channels were presented in [8,9,10].

IV. Related Work

A major complication for adopting threshold cryptography to asynchronous distributed systems is that many early protocols are not robust and that rely heavily on synchronous broadcast channels.

Shoup and Gennaro [4] present the first *robust threshold cryptosystem* that is also non-interactive, and as such integrates well into asynchronous communication model. Moreover, it is the first practical threshold cryptosystem that is provably secure against chosen ciphertext attack in the random oracle model. In the random oracle model cryptographic hash functions are replaced by a random oracle. This model was used informally by Fiat and Shamir [11] and later was rigorously formalized and more fully exploited in Bellare and Rogaway [12]. In the random oracle model the proof of security is viewed as “strong evidence“ that the scheme is actually secure in the “real world“. Authors presented and analyzed two schemes, which are based on the hardness of the Diffie-Hellman problem.

The *threshold RSA signature* scheme of Shoup [13] is unforgeable and robust in the random oracle model, assuming the RSA problem is hard. Signature share generation and verification is completely non-interactive.

First implementations of threshold signatures in asynchronous networks without random oracles are RSA signature schemes by Gennaro, Halevi and Rabin [14] and by Cramer and Shoup [15], which are based on strong RSA assumption.

The first practical *verifiable secret sharing* protocol for asynchronous networks together with a *proactive refresh protocol* is proposed by Cachin et al. [16]. The authors propose a model of *asynchronous proactive network* that extends an asynchronous network by an abstract timer that is accessible to every server. The timer defines the phase of a server locally. They assume that the adversary corrupts up to *t* different servers who are in the same local phase. Uncorrupted servers who are in the same local phase use private authenticated channels for communication. Message delay in such a channel must be no longer than the local phase lasts. Otherwise the message is lost. A proactive cryptosystem refreshes the shares of the secret key at the beginning of every phase. The liveness of the cryptosystem is based on the assumption that the adversary delays messages of the refresh protocol for no longer than the phase lasts. Otherwise the secret key may become inaccessible. This assumption seems reasonable because a phase typically lasts much longer than the maximal delay of a message in the real-world network. The *proactive refresh protocol* relies on a discrete logarithm-based verifiable secret sharing that is similar to Pedersen’s scheme [17]. The servers exchange two asynchronous rounds of messages to reach *agreement* on the success of the sharing. Agreement is achieved by using a randomized asynchronous multi-valued Byzantine agreement primitive [18]. Cachin et al. [16] left open the question of how proactive secure message transmission could be implemented.

A protocol for proactive secure message transmission over an asynchronous network is presented in [19]. The authors specify proactive secure message transmission in terms of an idealized service that has simple deterministic semantics and hides cryptographic objects from its interfaces. Additionally, a real implementation is proposed and proved to be at least as secure as the ideal service. The solution relies on a hardware

assumption, i.e., a secure co-processor that cannot be corrupted by the adversary.

The first purely asynchronous group key exchange protocol that tolerates a minority of servers to crash is presented in [20]. A group of servers communicate over an asynchronous network to establish a common session key such that anyone outside the group that can only observe the network traffic cannot learn this key. Such a key can later be used to achieve multicast message confidentiality or data integrity. The protocol consists of the following two stages. In the first stage, the group members exchange keying information using two communication rounds. In the second stage, they execute consensus protocol to select the contributions from the first stage where the session key is computed. The protocol may use randomized asynchronous consensus in the fully asynchronous model or a consensus protocol in the asynchronous model augmented with a failure detector. It is shown that any group exchange protocol among n servers that tolerates $t>0$ servers to crash can only provide forward secrecy if the adversary occupies less than $n-2t$ servers and the presented protocol achieves this bound.

CODEX (COrnell Data Exchange)[21] is a distributed service for storage and dissemination of secret keys that uses an approach to building distributed services that are both fault-tolerant and attack tolerant. This approach includes asynchronous model of execution, which makes the system resistant to denial of service attacks. Byzantine quorum systems are used for storing the state, ensuring consistency among the servers, and proactive secret sharing with threshold cryptography implement confidentiality and authentication of service responses.

The storage and transmission of data files in distributed systems gives rise to significant security and reliability problems. Information dispersal algorithms store files by distributing them among a set of servers in a storage efficient way. The authors in [22] introduce the problem of verifiable information dispersal in an asynchronous network, where up to one third of servers as well as an arbitrary number of clients might have Byzantine faults. Consistency of the stored information is ensured by verifiability. The secrecy of the stored data is guaranteed with respect to an adversary that may mount adaptive attacks.

V. Future Research

The authors in [23] describe some research subjects that are important in the future development of distributed cryptography. For instance, there exist many situations in which general structures instead of threshold structures are required. Moreover, it is necessary to find new public key cryptosystems for non-threshold structures. The design of distributed cryptosystems with non-threshold access structure is closely related to the problem of performing multiparty computation on general access structures. The main question is how to find efficient linear secret sharing schemes with the multiplicative property and very little is known about that.

VI. Conclusion

We have presented main results in the field of threshold cryptography under the asynchronous model of distributed computing. We started with fundamental definitions, then pointed out the meeting place of cryptography and distributed computing, and finally presented several solutions with concluding remarks concerning future research.

References

[1] Y. Desmedt, "Threshold cryptography," *European Trans. on Telecommunicatons*, 5(4), pp. 449-457, July-August 1994, (Invited paper).

[2] Y. Desmedt, "Some Recent Research Aspects of Threshold Cryptography," *In Eiji Okamoto, George Davida, and Mashiro Mambo, editors, Information Security, The 1st International Workshop, ISW'97,* Tatsunokuchi, Ishikawa Japan, September 17-19, 1997.

[3] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," *In G. Brassard, editor, Proc. CRYPTO 89*, pages 307-315. *Springer-Verlag, 1990, Lecture Notes in Computer Science No. 435.*

[4] V. Shoup and R. Gennaro, "Securing threshold cryptosystem against chosen ciphertext attack," *Proc. EURO-CRYPT '98, LNCS 1403*, 1998.

[5] R. Canneti, R. Gennaro, A.Herzberg, and D. Naor, "Proactive security: Long-term protection against break-ins," *RSA Laboratories' CryptoBytes, vol. 3, no.1,* 1997.

[6] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable secret sharing and achieving simultaneity in the presence of faults," *Proc. 26th IEEE Symp. on Found. of Computer Science*, pages 383-395, 1985.

[7] O. Goldreich, "Cryptography and Cryptographic Protocols," *Distributed Computing*, v.16, n.2-3, p.177-199, September 2003.

[8] M. Ben-Or, R. Canetti and O. Goldreich, "Asynchronous Secure Computation," *25th ACM Symposium on the Theory of Computing*, pages 52-61, 1993.

[9] M. Ben-Or, B. Kelmer and T. Rabin, "Asynchronous Secure Computations with Optimal Resilience," *13th ACM Symposium on Principles of Distributed Computing*, pages 183-192, 1994.

[10] D. Dolev, C. Dwork, O. Waarts, and M. Yung, "Perfectly secure message transmission," *Journal of the ACM*, Vol. 40(1), pages 17-47, 1993.

[11] A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems", *Advances in Cryptology-Crypto '86,* Springer LNCS 263, pages 186-194, 1987.

[12] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols", *First ACM Conference on Computer and Communications Security*, pages 62-73,1993.

[13] V. Shoup, "Practical threshold signatures," *Proc. EURO-CRYPT 2000, LNCS 1087*, pp.207-220, 2000.

[14] R. Gennaro, S. Halevi, and T. Rabin, "Secure hash-and-sign signatures without the random oracle," *Proc. EUROCRYPT '99,* pp. 123-139, Springer, 1999.

[15] R. Cramer and V. Shoup, "Signature schemes based on the strong RSA problem," *ACM Transactions on Information and System Security*, vol. 3, no. 3, pp. 161-185, 2000.

[16] C. Cashin, K. Kursawe, A. Lysyanskaya, and R. Strobl, "Asynchronous Verifiable Secret Sharing and Proactive Cryptosystems," *Proc. 9th ACM Conference on Computer and Communications Security (CCS)*, pages 88-97, 2002.

[17] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," *In J. Feigenbaum, editor, CRYPTO '91, volume 576 of LNCS*, pages 129-140, Springer, 1992.

[18] C. Cashin, K. Kursawe, F. Petzold, and V. Shoup, "Secure and efficient asynchronous broadcast protocols," *Advances in Cryptology-Crypto 2001, 2001*.

[19] M. Backes, C. Cashin, and R. Strobl, "Proactive Secure Message Transmission in Asynchronous Networks," *Proc. 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003)*, pages 223-232, July 2003.

[20] C. Cashin, and R. Strobl "Asynchronous Group Key Exchange with Failures," *Proc. 23rd ACM Symposium on Principles of Distributed Computing (PODC 2004)*, pages 357-366, July 2004.

[21] M. A. March, F. B. Schneider, "CODEX: A Robust and Secure Secret Distribution System," *IEEE Transactions on Dependable and Secure Computing*, January-March 2004 (Vol.1, No.1) pp. 34-47.

[22] C. Cashin, S. Tessaro, "Asynchronous Verifiable Information Dispersal," *Research Report RZ 3569, IBM Research*, December 2004.

[23] Research Group on Mathematics Applied to Cryptography, "Some trends for future research in distributed cryptography," *Stork Cryptography Workshop: Towards a Roadmap for Future Research*, November 26-27, 2002.