Does Proactive Secret Sharing Perform in Peer-to-Peer Systems?

Nicolas C. Liebau[1], Andreas U. Mauthe[2], Vasilios Darlagiannis[3], and Ralf Steinmetz[1]

1 **Multimedia Communications Lab, Technische Universität Darmstadt**
2 **Computing Department, Lancaster University**
3 **Informatics and Telematics Institute, Centre for Research and Technology Hellas**

Abstract

Trustworthy applications in fully decentralized systems require a trust anchor. This paper describes how such an anchor can be implemented efficiently in p2p systems. The basic concept is to use threshold cryptography in order to sign messages by a quorum of peers. The focus is put on advanced mechanisms to secure the shares of the secret key over time, using proactive secret sharing. This mechanism was researched in context of the token-based accounting scheme.

Keywords and phrases peer-to-peer, proactive secret sharing

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.239

1 Introduction

Trustworthy applications require an trust anchor, a trustworthy foundation that security mechanisms can employ. Typically in IT systems a trust anchor is a trusted entity with its associated public and private key. This key pair is used as focal point with which the security of a mechanism can be checked.

In p2p systems a single trusted entity does not exist by definition. Thus, if a p2p application requires trust, an alternative is to use a group of peers as the trust anchor. Such a so called quorum delivers trust to the application by probability - the probability that in the quorum fraudulent behavior will not prevail. E.g., in a quorum requiring unanimous judgment a quorum size of 17 guarantees trust with a probability of 99.999%, if there are up to 50% fraudulent peers in the system [7]. Therefore, threshold cryptography has been researched as a way to implement such unanimous quorum judgment. This would build a decentralized trust anchor.

A successful implementation of threshold cryptography in a p2p system consists of several elements. Threshold cryptograhy splits a secret key s into n shares and requires only t shares to create a signature of a message. This is called a (t, n)-threshold scheme.

The focus of this paper is on mechanisms that secure the key shares over time. Peers get compromised over time and their key shares get know to fraudulent peers. If a fraudulent peer collects the knowledge of t shares it can forge system signatures. Proactive Secret Sharing (PSS) was invented to deal with this issue. It introduces time periods and invalidates all key shares at the end of a period by updating them and recovering shares for peers that have been corrupted.

Within the token-based accounting scheme (TbAS) [7, 8] PSS has been employed to ensure the system's long term trustworthiness. However, updating a complete p2p system seems prohibitive expensive. Among other things, this is the reason for introducing so called

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 239–244

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

trusted peers in TbAS. This is a subset of peers that have been selected as share owner due to their trustworthiness using a reputation mechanism.

This paper proves by example of the TbAS that PSS is not prohibitive expensive for applying it to p2p systems in order to build a decentralized trust anchor.

2 Related Work

Threshold cryptography has been suggested to be used in p2p systems by several authors. In [14, 15, 13] the costs of applying threshold cryptography in p2p systems is evaluated; however, focus is on the signature process. The costs of maintaining a distributed secret over time is not shown. In [2] the application of PSS to MANET is discussed but not evaluated. In [11] a DRM system based on PSS is presented, however a thorough evaluation of the created traffic by the PSS mechanism is missing. An evaluation of PSS for up to 100 peers is given presented in [10], assuming that all peers are updated.

3 Cryptographic Background

In order to allow building a trust anchor threshold cryptography (see e.g. [3, 6, 17, 4]) offers the required mechanisms;

3.1 Threshold Cryptography

When selecting a threshold cryptography scheme attention must be paid to the secret sharing mechanism. In additive secret [6] a secret s is split into n shares, and share s_n is computed from the other $n-1$ parts. When a secret key is created and shared among peers, all shares have to be created at the same time. Additional shares cannot be created later. Therefore, additive secret sharing is not applicable to p2p system, where membership is dynamic. In [16] Shamir presented polynomial secret sharing, that does not have this drawback. Here, the shares are calculated by using a polynomial $f(x)$ of degree $t-1$, where the secret key is $s = f(0)$. Accordingly, only threshold schemes based on polynomial secret sharing are considered for building a distributed trust anchor.

There are several challenges to overcome in order to apply threshold cryptography to p2p systems (see [8]); The URSA scheme[9] and BLS scheme [1] are build on polynomial secret sharing and fulfill all remaining requirements.

3.2 Proactive Secret Sharing

This paper focuses Proactive Secret Sharing (PSS) [5] applied in p2p systems. PSS schemes were introduced to protect long-lived shares of cryptographic keys. The principle of PSS is to introduce time periods. In each time period the shares of the shareholders are updated by adding new polynomials $g(x)$ with $g(0) = 0$. That is, all key shares change with an update period, however the secret key remains unchanged. Further, peers with corrupted shares can be recovered; that is a new share for the new sharing polynomial is calculated for them is a distributed fashion by a group of k updated peers. Using recovery, also new peers in the system can get assigned a new share. For details about the protocols used in the TbAS see [8].

Both considered schemes, Threshold BLS and URSA, require the same message flow for a share update or a share recovery. When the shares are distributed, each peer requires an individual share. Accordingly, updating large systems seems to be expensive traffic-wise.

Table 1 Experiments for System-key Maintenance

$L = 99,99\%$				$L = 99,999\%$			
#	T	t	β	#	T	t	β
1.1	100	13	26	2.1	100	15	30
1.2	500	14	28	2.2	500	17	34
1.3	1000	14	28	2.3	1000	17	34
1.4	2000	14	28	2.4	2000	17	34

Legend: T: Number of trusted peers, t: quorum size, β: Size of update group

In [8] different update strategies are evaluated. In [9] a scheme is suggested where only a specific ratio of peers is updated and the remaining peers recover their share. This has the advantage that the knowledge of all trusted peers' IDs is not required (see [8]). However, an evaluation for large p2p systems was not performed. In context of TbAS this Limited Update and Self-Initialization mechanism was evaluated with the target of building highly trustable p2p mechanisms.

4 Simulation of Key Management Traffic

In order to simulate key management, two parts of proactive secret sharing, namely the update phase and the recovery phase were implemented in PeerfactSim.KOM [12]. The objective of this simulation was to assess the traffic created when all trusted peers receive an updated key share of the system-wide private key.

4.1 Experiments

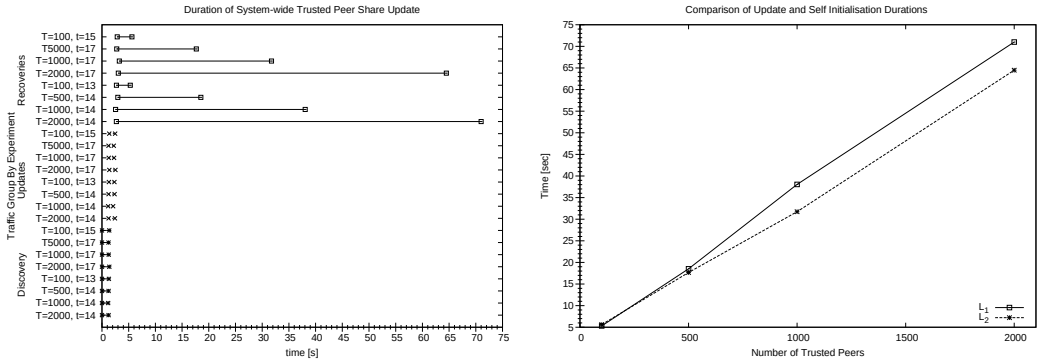
When the proximity group B is found the update shares have to be distributed among the peers in this group. This traffic is fully deterministic, because only direct communication between the group members is applied. Therefore, the general concept of the experiment is to perform the update using only one update polynomial by one peer. This allows an exact analysis of the created traffic that can be easily extrapolated to the use of more update polynomials.

the proximity group B's size β must have at least the size t in order to enable recoveries by that group. Within the simulation, the update group size β is set to $\beta = 2t$ in order to achieve redundancy.

In order to evaluate the scalability of the update and self initialization scheme, different numbers of peers between $T = 100$ and $T = 2000$[1] to be updated and recovered as well as different Trust Levels L (determined by the threshold t) will be simulated[2]. Table 1 summarizes the executed experiments. In all experiments we use a key length of 1024 bit.

[1] Within TbAS scheme threshold cryptography is executed by a subgroup of peers that are called trusted peers. Even for very large p2p systems a number of $T = 2000$ trusted peers is sufficient. See [7, 8].

[2] The computation of required threshold depending on the desired system's Trust Level L, number of peer T, and ratio of good peer p_g was presented e.g .in [7, 8]. For the experiments a ratio of good peer $p_g = 0.5$ is assumed.



(a) Durations by partial mechanism **(b)** Comparison of durations by system size

Figure 1 Durations for update and self-initialization

4.2 Results

The update and self initialization mechanism is a very deterministic process. The simulation does not require any probability distributions. Accordingly, the confidence intervals are very small and cannot be seen in the figures.

4.2.0.1 Duration of Update and Self initialization Process

First, the time required for completely updating the trusted peer system is looked at. Figure 1 a shows the processes' duration.[3] The discovery of the initial update group requires between 1.16 seconds and 1.36 seconds. The difference is due to message transmission delays. The update phase required a bit less time and needed between 0.98 seconds and 1.18 seconds. the reason is that lookup messages might require several hops on the way towards the peer, where update requires direct communication. The recovery phase required 2.58 seconds for systems with $T = 100$ trusted peers and a quorum size of $t = 13$ and 68.3 seconds for a system size of $T = 2000$ and a quorum size of $t = 14$ trusted peers. The average time for a quorum size of $t = 17$ trusted peers is here a bit lower, which indicates that the quorum size has no influence on the duration of the recovery process. The reason is that the number of communication steps required within a recovery of one trusted peer is constant.

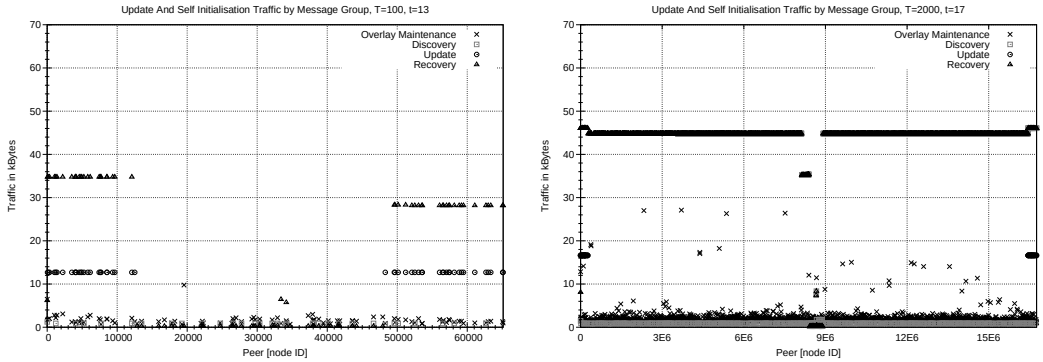
The difference in the duration stems from the different sizes of the beginning start-up group. With a higher Trust Level, the update group in the beginning is larger; updating peers happens in parallel. Therefore, the update duration does not increase with the update group size. However, with a larger update group size less recoveries are required. Therefore, for $L_2 = 99.999\%$ the update duration is shorter.

4.2.0.2 Traffic Generated by Update and Self initialization

The traffic is observed on a per-peer basis. Figure 2 shows the traffic generated for the smallest and largest simulation setzup. The x-axis represents the peers' ID.[4]

It can be observed that the update group is symmetrically located around the peer ID 0. Especially in the graphs for large system sizes with $T = 2000$, the recovery traffic

[3] Note that we did not simulate computation time. The durations all stem from message transfer.
[4] For $T = 2000$ the ID space was increased in order to enable complete message logging. Therefore, peer IDs are distributed from 0 to 2^{24}.



(a) Traffic by message type, T=100, t=13 **(b)** Traffic by message type, T=2000, t=17

Figure 2 Update and self initialization traffic by message type

is almost equal for all peers. Only at the middle of the graph is the traffic distinctively lower, because here the two recovery fronts meet. Recovery traffic is the major source for traffic, although it is below 60 kBytes in total per peer. It can be seen that recovery traffic and overlay maintenance traffic have the lowest proportions of traffic load. Overall, a very even distribution of traffic load over the trusted peers is achieved by the update and self initialization mechanism.

When observing the main source for traffic increase, it is obvious that the quorum size has an influence on it, however the system size influence seems minimal. It can be seen that there are very few peers with distinctively higher loads. The maximum load for the complete update and self initialization process is 240.13 kBytes, which happened in all experiments with $t = 17$. Assuming for trusted peers a DSL-connection with 128 kBit upload, a trusted peer would need 15.37 seconds to send this traffic. Accordingly, even the very few trusted peers with the maximum load do not get overloaded by the update and self initialization process. The statistics show that there are only very few peers with a traffic load above 100 kBytes.

5 Conclusion

This paper focused on a specific issue when applying threshold cryptography to p2p systems. The key shares have to be protected over time. Proactive secret sharing (PSS) is designed for that purpose. PSS introduces time periods and provides the ability to update key shares and recover corrupted and lost key shares at the end of a time period. Also new key shares for new peers can be created. However, these mechanisms seem to be expensive and unattractive for large p2p system. In this paper we proved the opposite. Updating a system of 2000 peers requires approximately and 71 seconds and the average load per peer is 49.10 kBytes.

Accordingly, threshold cryptography based p2p systems can be secured with PSS. Update phases can be performed once a day. Peers not participating in an update will be recovered when they join again. This basic finding can be employed to build decentralized trust anchors for new trustworthy p2p mechanisms, like the token-based accounting scheme [7, 8].

References

1 Alexandra Boldyreva. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In *PKC '03: Proceedings of the 6th*

International Workshop on Theory and Practice in Public Key Cryptography, volume 2567 of *LNCS*, pages 31–46, London, UK, 2003. Springer-Verlag.

2 Vanesa Daza, Javier Herranz, Paz Morillo, and Carla Ràfols. Cryptographic techniques for mobile ad-hoc networks. *Computer Networks*, 51:4938–4950, 2007.

3 Y. Frankel, P. Gemmell, P. D. MacKenzie, and Moti Yung. Optimal-Resilience Proactive Public-Key Cryptosystems. In *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, page 384, Washington, DC, USA, 1997. IEEE Computer Society.

4 Amir Herzberg, Markus Jakobsson, Stanislaw Jarecki, and Hugo Krawczyk. Proactive Public Key and Signature Systems. In *Proceedings of ACM Conference on Computer and Communications Security*, pages 100–110, 1997.

5 Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive Secret Sharing Or: How to Cope With Perpetual Leakage. In *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '95)*, volume 963 of *LNCS*, pages 339–352, London, UK, 1995. Springer-Verlag.

6 Stanislaw Jarecki and Nitesh Saxena. Further Simplifications in Proactive RSA Signatures. In *Proceedings of Theory of Cryptography Conference'05*, pages 510–528, 2005.

7 Nicolas Liebau, Vasilios Darlagiannis, Oliver Heckmann, and Andreas Mauthe. *Peer-to-Peer Systems and Applications*, volume 3485 of *LNCS*, chapter Accounting in Peer-to-Peer Systems, pages 547 – 566. Springer-Verlag, 2005.

8 Nicolas C. Liebau. *Trusted Accounting in Peer-to-Peer Environments - A Novel Token-based Accounting Scheme for Autonomous Distributed Systems*. PhD thesis, Technische Universität Darmstadt, Germany, 2008.

9 Haiyun Luo, Jiejun Kong, Petros Zerfos, Songwu Lu, and Lixia Zhang. URSA: Ubiquitous and Robust Access Control for Mobile Ad Hoc Networks. *IEEE/ACM Transactions on Networking*, 12(6):1049–1063, 2004.

10 Haiyun Luo, Petros Zerfos, Jiejun Kong, Songwu Lu, and Lixia Zhang. Self-Securing Ad Hoc Wireless Networks. In *ISCC '02: Proceedings of the 7th International Symposium on Computers and Communications (ISCC'02)*, page 567, Washington, DC, USA, 2002. IEEE Computer Society.

11 Ling Ma, Shouxun Liu, and Yongbin Wang. A DRM model based on proactive secret sharing scheme for p2p networks. In *Cognitive Informatics (ICCI), 2010 9th IEEE International Conference on*, pages 859 – 862, 2010.

12 Multimedia Communications Lab – Technische Universität Darmstadt. PeerfactSim.KOM. http://peerfact.kom.e-technik.tu-darmstadt.de/, 2007.

13 Nitesh Saxena. *Decentralized Security Services*. PhD thesis, University of California, Irvine, 2006.

14 Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi. Admission Control in Peer-to-Peer: Design and Performance Evaluation. In *SASN '03: Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 104–113, New York, NY, USA, 2003. ACM Press.

15 Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi. Threshold Cryptography in P2P and MANETs: The Case of Access Control. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 51(12):3632–3649, 2007.

16 A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, November 1979.

17 Victor Shoup. Practical Threshold Signatures. In *Proceedings of Eurocrypt 2000*, 2000.