Network Protocols and Algorithms
ISSN 1943-3581
2009, Vol. 1, No. 1

Dynamic Threshold Cryptosystem without Group Manager

Andreas Noack
Chair for Network and Data Security
Ruhr-University Bochum, 44780 Germany
Tel: (+49) (0)234 32 – 26742 E-mail: andreas.noack -at- rub.de

Stefan Spitz
Workgroup for Integrated Information Systems
Ruhr-University Bochum, 44780 Germany
Tel: +49 (0)234-32-22476 E-mail: stefan.spitz -at- rub.de

Abstract

In dynamic networks with flexible memberships, group signatures and distributed signatures are an important problem. Dynamic threshold cryptosystems are best suited to realize distributed signatures in dynamic (e.g. ad-hoc) networks. Without a group manager or a trusted third party even more flexible scenarios can be realized.

Gennaro et al. [1] showed, it is possible to dynamically increase the size of the signer group, without altering the public key. We extend this idea by removing members from the group, also without changing the public key. This is an important feature for dynamic groups, since it is very common, e.g. in ad-hoc networks that members leave a group.

Gennaro et al. used RSA and bi-variate polynomials for their scheme. In contrast, we developed a DL-based scheme that uses ideas from the field of proactive secret sharing (PSS). One advantage of our scheme is the possibility to use elliptic curve cryptography and thereby decrease the communication and computation complexity through a smaller security parameter.

Our proposal is an efficient threshold cryptosystem that is able to adapt the group size in both directions. Since it is not possible to realize a non-interactive scheme with the ability to remove members (while the public key stays unchanged), we realized an interactive scheme whose communication efficiency is highly optimized to compete with non-interactive schemes. Our contribution holds against passive and active information theoretic adversaries.

Keywords: Public-key cryptosystem, threshold cryptosystem, dynamic adding, dynamic removing, dynamic membership, secret sharing, proactive

1. Introduction

Anonymous and autonomous group oriented cryptosystems have several basic requirements such as the anonymity of group members to outsiders and no need for a trusted third party. Additionally it is required to need at least a distinct number of cooperating group members to accomplish group operations, in which the secret key of the group is involved.

Furthermore, to call the system autonomous, it is mandatory to be able to add and remove members *without* a trusted third party. A subset of the group decides on adding new or removing current members, whereby the public and secret key of the group remains untouched, even when the group size changes.

Applications for those cryptosystems are firstly dynamic networks like mobile ad-hoc networks or mesh networks, in which mobile nodes constantly leave and join. Threshold signatures, threshold decryption and secure routing are examples for the cryptographic usage.

Second, it can be used in applications distributed among a network, for example distributed intrusion detection systems (IDS). In distributed IDS, many network sensors communicate jointly with an automatic or manual warning system.

Scenarios without a trusted third party become more and more important, since the network complexity increases due to new types of networks (i.e. ad-hoc, mesh networks) and a centralized TTP would introduce a single point of failure.

We considered threshold and identity based cryptography to find an appropriate solution for these scenarios.

1.1 Related Work

In 1984, Shamir [2] proposed the concept of id-based cryptography. In this concept, a user is identified by a publicly known parameter such as an e-mail address, IP address or name. This identifier can be used in conjunction with the public key of the group to encrypt data destined to or verify signatures originating from this user. A trusted third party who generates secret keys that correspond to the public key of the group *and* the ID of the user is necessary.

While Shamir provided the id-based digital signature scheme, Boneh and Franklin [3] as well as Cocks [4] independently introduced schemes for id-based encryption in 2001 which are based on Weil pairings respectively the problem of quadratic residues.

In 2003 Libert and Quisquater [5] introduced a threshold cryptosystem based on pairings with the ability to efficiently revocate users within the group. To accomplish this task they used the idea of a so-called *semi-trusted mediator* (mostly a synonym for TTP) proposed by

Boneh, Ding, Tsudik and Wong [6] in 2001.

An alternative to id-based cryptography is the idea of using threshold schemes. In 1979, Shamir [7] presented the idea of "How to Share a Secret". In his scheme, the secret is divided and distributed among several entities. This secret can be reconstructed when some of the entities work together. Desmedt and Frankel [8] used this idea to design a threshold cryptosystem based on ElGamal in 1989.

During 1991, Pedersen [9] presented another threshold scheme based on Shamir's idea and the ElGamal cryptosystem without a TTP or group manager. An additional property of this scheme is the verifiability of all member shares (VSS - Verifiable Secret Sharing).

In 2005, Saxena, Tsudik and Yi [10] presented a scheme with the option of dynamically adding members to the group. Their scheme uses bi-variate polynomials, which allowed them to design a non-interactive threshold scheme. The key establishment can be performed by a set of founding members using the method of joint secret sharing [11], while adding members is possible through a subset of already established members of the group.

Another approach on threshold schemes is based on RSA. In 1991 Desmedt and Frankel [12] initiated the study of using RSA for threshold schemes. Similar to the ElGamal-based scheme of [8], adding or removing a member is not possible in their scheme.

Gennaro et al. [13] introduced some ideas to provide robustness to RSA threshold schemes in 1996, with Shoup [14] presenting a robust RSA threshold signature scheme in 1999.

Recently, during 2008 Gennaro et al. [1] developed a dynamic RSA threshold scheme. They utilize bi-variate polynoms for adding new members while being dependent on a trusted third party in the initial key distributing phase. In addition to the dynamic member adding, their scheme provides robustness and is non-interactive.

In 1995, Herzberg et al. [15] developed a method to increase the security of a (k, n)-threshold scheme by decreasing the time window during which an adversary must compromise $\geq k+1$ shares of the secret. In this proactive secret sharing scheme (PSS) the shares of all members are periodically renewed.

1.2 Our Contribution

The threshold scheme [5] uses id-based cryptography and provides the possibility to add and remove members dynamically, while being dependent on a TTP. Scheme [10] is based on ElGamal and allows the adding of members without the help of a TTP. In the RSA-based scheme [1], the adding of members requires a TTP without providing an option of removing members. Table 1 shows the comparison of these schemes (and their predecessors) with our scheme.

Table 1. Comparison of referenced schemes

Scheme	**Establish Group**	**Add Member**	**Remove Member**
Desmedt, Frankel[12] (RSA)	○	-	-
Genarro et al. [1] (RSA)	○	●	-
Boneh, Franklin [3] (ID)	○	○	(○)
Libert, Quisquater [5] (ID)	○	○	○
Pedersen [9] (ELG)	●	-	-
Saxena et al. [10] (ELG)	●	●	-
Our Scheme (ELG)	●	●	●

ELG=ElGamal-based, RSA=RSA-based, ID=id-based

● without TTP, ○ with TTP, (○) with CRL/Timestamp, - not possible

To the best of our knowledge, there is currently no threshold cryptosystem (neither id-based, ElGamal-based nor RSA-based) which allows the removing of members without a TTP. Certificate revocation lists could be used to replace a TTP, but this possibility will not be considered due to the unbounded size of those lists and their impact to the anonymity property.

Our contribution is an ElGamal-based anonymous and autonomous threshold scheme based on techniques derived from the share renewal method in PSS. Predefining a maximum number of group members is no longer necessary either.

2. Threshold Scheme without Group Manager

We have searched for a group oriented cryptosystem that has the following properties: it is autonomous, anonymous and provides security against inside and outside adversaries.

ID-based systems do not accomplish the autonomous and anonymous property. A group manager is necessary in the known schemes based on pairings and furthermore no anonymity is present due to the fact that each party has its own public key. Group signature schemes need a group manager for the possibility to identify signers (signer ambiguous) and for adding or removing members.

We came to the conclusion to deploy threshold schemes, since a group manager is not mandatory here and anonymity can be provided as well.

A (k, n)-threshold scheme has n members from which at least $k + 1$ members are necessary to recover the group secret. This scheme is extended to a public key cryptosystem with a shared secret key SK and a single public key PK.

Generating and distributing the public/secret key pair can be done by a TTP or a distributed key generation. A TTP generates the (PK, SK) pair, splits the secret key into n

shares and distributes them securely to each member as used in Shamir's secret sharing [7]. Our introduced scheme uses the second possibility where all members cooperate in the key generation.

2.1 Distributed Key Generation

In this section, we show how a group $U = \{P_1 \dots P_n\}$ can cooperate to establish a public key $PK = (p, q, g, A)$ and corresponding secret key $SK = (a_0)$, such that at least $k + 1$ members are needed to utilize SK. The distributed key generation is divided into two operations, the key generation and distribution. Prior to key generation, all members P_i need to agree on the public parameters p, q, g such that:

$$p, q \in \mathbb{P}\colon\ p = 2 \cdot q + 1$$
$$g \in \mathbb{Z}_p^*\colon\ ord(g) = q$$

After agreeing on the public parameters and the desired threshold value k, each member P_i of group U generates a key pair for the El-Gamal encryption scheme as follows:

$$1.)\ u_{i,0} \in_R \mathbb{Z}_q^*$$
$$2.)\ A_i \coloneqq g^{u_{i,0}}\ mod\ p$$

with A_i being a part of the public key parameter A. The value $u_{i,0}$ is called *member secret* of P_i. It represents a part of the group's secret key a_0 and is distributed among all members of U. This is done by creating a k-resilient (k, n) secret sharing scheme with the idea of Shamir[7].

$$1.)\ \{s_1, \dots, s_k\} \in_R \mathbb{Z}_q^*$$
$$2.)\ f_i(x) = s_k x^k + s_{k-1} x^{k-1} + \dots + s_1 x^1 + u_{i,0}\ mod\ q$$
$$3.)\ u_{i,j} \coloneqq f_i(j), \qquad \forall j: P_j \in U$$
$$4.)\ \text{Send: } \langle \mathrm{A_i}, \mathrm{u_{i,j}} \rangle \text{ to } P_j \text{ (confidentially)}, \qquad \forall\ P_j \in U$$

Each member P_i chooses a random polynomial $\deg(f_i) = k$ and computes n *member shares* $u_{i,j} = f_i(j)$ of its own secret polynomial. Then the member shares, as well as the values A_i, are sent confidentially to each P_j. After the distribution, each member computes its so-called *secret share* and public key parameter A.

$$1.)\ A = \prod_{i=1}^{n} A_i\ mod\ p$$
$$2.)\ a_i = \sum_{j=1}^{n} u_{j,i} \equiv \sum_{j=1}^{n} f_j(i)\ mod\ q$$

Each P_i is now in possession of $PK = (p, q, g, A)$ and a secret share a_i of $SK = (a_0)$. The secret share a_i is the function evaluation of $f(x) = \sum_{j=1}^{n} f_j(x)\ mod\ p$ at position i (homomorphism property).

2.1 Adding Members

The adding of a specific user P_{n+1} to a pre-established group U with n members is explained in detail within this section. The following conditions must hold when a user is added to the group.

- The public key $PK = (p, q, g, A)$ and the secret key $SK = (a_0)$ remain unchanged.
- The secret share a_{n+1} and polynomial f_{n+1} is only known to P_{n+1}.
- The new a_{n+1} of P_{n+1} is a valid and fully functional secret share of the group U.

We adapt the share renewal technique used in PSS[15] to implement an efficient adding and removing of members. While adding, $k+1$ members split off a part of their secret and share this part with the new user. Removing a member is done by computing and redistributing the member's secret to some remaining members, as shown in Section 2.2. The adding of a user is done in three phases.

2.1.1 Phase 1a - Secret Sharing

During phase 1a, $k+1$ members of U form a so called *helper group* U_h. These members cooperate in the generation of a new member secret $u_{n+1,0}$ for P_{n+1}. For this, all $P_i \in U_h$ cede a part of their member secret $u_{i,0}$ and submit it secretly to P_{n+1}. A new polynomial is generated and the differences between the values generated from the new and the old polynomial are also distributed secretly. The details of this process executed by all $P_i \in U_h$ are shown below:

$$\begin{aligned}
&1.)\ share_i \in_R \mathbb{Z}_q^* \\
&2.)\ u_{i,0} = u_{i,0} - share_i \\
&3.)\ s_t \in_R \mathbb{Z}_q^*, \qquad \forall t \in \{1, \dots, k\} \\
&4.)\ f_i'(x) = s_k x^k + s_{k-1}x^{k-1} + \cdots + s_1 x^1 + u_{i,0} \bmod q \\
&5.)\ \delta_i[j] = f_i'(j) - f_i(j) \bmod q, \qquad \forall j : P_j \in U \\
&6.)\ f_i(x) = f_i'(x)
\end{aligned}$$

Each member $P_i \in U_h$ is now in possession of a new $f_i(x)$ and a new member secret $u_{i,0}$. All P_i then send (confidentially) $\langle share_i,\ \delta_i,\ u_{i,n+1}\rangle$ to P_{n+1}.

2.1.2 Phase 1b - Share Computation

In this phase, the member shares $u_{i,n+1}$ $(\forall i \in U)$ for the new member P_{n+1} are computed. During the distributed key generation all n members were needed to compute their member shares, but with the help of the Lagrangian interpolation only $k+1$ members need to cooperate to generate these n shares. The Lagrangian interpolation is defined as:

$$L(\alpha, \beta) \stackrel{\text{def}}{=} \prod_{\gamma \in U, \gamma \neq \beta} \frac{\alpha - \gamma}{\beta - \gamma}$$

The communication progress is similar to a chain reaction. Without loss of generality, we start with member P_1, incrementing step-wise until P_{k+1} is reached. The necessary $k+1$ members are allowed to be in U_h. Phase 1a and phase 1b can be run simultaneously, since the operations do not depend on each other. P_1 computes:

$$1.)\ r \in_R \mathbb{Z}_q^*$$
$$2.)\ \omega_j := u_{j,1} \cdot L(n+1,j) + r \bmod q, \qquad \forall j: P_j \in U \setminus U_h$$
$$3.)\ \Omega = \{\omega_1|\omega_2| \dots |\omega_j\}$$

The array Ω contains a subset of all computed member shares of P_i, $\forall i \in U \backslash U_h$ for user P_{n+1} which is sent (confidentially) to the next member within the chain. It is necessary that the second user in the chain is a member of the group U_h, so that P_2 is not able to recover the used r. In addition, r is sent (confidentially) to P_{n+1}. Now, members P_i, $\forall i \in \{2, \dots, k+1\}$ proceed with:

$$1.)\ \omega_j = \omega_j + \left(u_{j,i} \cdot L(n+1,j)\right) \bmod q, \qquad \forall j: P_j \in U \setminus U_h$$
$$2.)\ Send\ \Omega\ to\ P_{j+1}(\text{confidentially})$$

The last member P_{k+1} sends Ω confidentially to P_{n+1}.

2.1.3 Phase 2 - Reconstructing Shares

At the start of phase 2 the (yet to be added) new user P_{n+1} receives the messages from phase 1a and phase 1b, hence $\langle share_i, \delta_i, u_{i,n+1}\rangle$, $\forall i \in U_h$ and $\langle\Omega\rangle$. When P_{n+1} has received all $h+1$ messages, the following is done:

$$1.)\ u_{i,n+1} = \omega_i - r, \qquad \forall i: P_i \in U \setminus U_h$$
$$2.)\ u_{n+1,0} = \sum_{i:P_i \in U_h} share_i \bmod q$$
$$3.)\ s_t \in_R \mathbb{Z}_q^*, \qquad \forall t \in \{1, \dots, k\}$$
$$4.)\ f_{n+1}(x) := s_k x^k + s_{k-1} x^{k-1} + \dots + s_1 x + u_{n+1,0} \bmod q$$
$$5.)\ a_{n+1} := \sum_{i:P_i \in U \cup P_{n+1}} u_{i,n+1} \bmod q$$
$$6.)\ \Delta = \{\delta_1| \delta_2| \dots |\delta_i\}, \qquad \forall i: P_i \in U_h$$
$$7.)\ \Lambda = \{u_{n+1,1}\} |u_{n+1,2}| \dots |u_{n+1,n}\}, \qquad with\ f_{n+1}(i) = u_{n+1,i}$$

Then P_{n+1} sends $\langle \Delta, \Lambda\rangle$ via broadcast (confidentially).

2.1.4 Phase 3 - Final Computation

During phase 3 each member P_i receives the broadcast from P_{n+1} and computes:

$$\begin{aligned}&1.)\ u_{j,i} = u_{j,i} + \delta_j(i) \mathit{mod}\ q, \qquad \forall j\colon P_j \in U_h\\ &2.)\ u_{n+1,i} = \Lambda[i]\\ &3.)\ a_i = \sum_{j=1}^{n+1} u_{j,i} \mathit{mod}\ q\end{aligned}$$

All members now possess the member share of P_{n+1} and the updated member shares from the members of U_h. Then, they compute their new secret share a_i adding the user P_{n+1} to the group.

2.2 Removing Members

Removing a member P_r is executed in three steps. First, the member secret $u_{r,0}$ is recovered using the Lagrangian interpolation by at least $k+1$ members. This secret is then shared between some members. From there on, the removing of a member is similar to the adding, with the details of each of the three phases shown below.

2.2.1 Phase 1 - Secret Recovery/Share Computation

The user P_r shall be removed from the group U. For this, $k+1$ members need to agree on the removing, forming the group U_{k+1}, with one of the members designated as P_z. Each of the members P_i, $\forall i \in \{1, \dots, k+1\}$ (w.l.o.g.) computes:

$$\omega_i := L(0,i) \cdot u_{r,i} \mathit{mod}\ q$$

With this, it is possible to reconstruct the secret of P_r by computing $u_{r,0} = \sum_i^{k+1} \omega_i$ later on.

A helper group U'_h with at least 2 members is formed. It is mandatory, that member P_z is in U'_h. Ideally, the remaining $h-1$ members are from U_{k+1} as well, which will reduce the number of communications necessary for this phase. Now, $h-1$ users P_i from U'_h will do the following operations and then send the results to the remaining user P_z of this group:

$$\begin{aligned}&1.)\ share_i \in_R \mathbb{Z}_p^*\\ &2.)\ u_{i,0} = u_{i,0} + share_i \mathit{mod}\ q\\ &3.)\ s_t \in_R \mathbb{Z}_q^*, \qquad \forall t \in \{1, \dots, k\}\\ &4.)\ f_i'(x) = s_k x^k + s_{k-1}x^{k-1} + \dots + s_1 x^1 + u_{i,0} \mathit{mod}\ q\\ &5.)\ \delta_i(x) = f_i'(x) - f_i(x) \mathit{mod}\ q\\ &6.)\ f_i(x) = f_i'(x)\end{aligned}$$

In comparison with the corresponding adding phase 1a, the computations differ in the computation of the differences (δ_i), because δ_i is now a polynomial. Therefore the data volume is much lower during removing phase, since the polynomial has only $k+1$ values (coefficients). The difference values during the adding phase consist of n values. Now, $\langle \omega_i,\ share_i,\ \delta_i \rangle$, $\forall j \in \{1,\dots,k+1\}$ and $\forall i \in U'_h$ is sent (confidentially) to P_z.

2.2.2 Phase 2 - Difference Transfer

During phase 2 all necessary computations for the removing of member P_r are executed. First, member P_z reconstructs the member secret of P_r by computing

$$u_{r,0} = \sum_{i=1}^{k+1} \omega_i \ mod\ q$$

Without loss of generality: $i = 1,\dots,k+1$. Then, P_z computes δ_z similar to the other users of the helper group U'_h.

$$\begin{aligned}
&1.)\ share_z = u_{r,0} - \sum_{i=1}^{h} share_i \ mod\ q\\
&2.)\ u_{z,0} = u_{z,0} + share_z \ mod\ q\\
&3.)\ s_t \in_R \mathbb{Z}_q^*, \qquad \forall t \in \{1,\dots,k\}\\
&4.)\ f_z'(x) = s_k x^k + s_{k-1}x^{k-1} + \cdots + s_1 x^1 + u_{z,0} \ mod\ q\\
&5.)\ \delta_z(x) = f_z'(x) - f_z(x) \ mod\ q\\
&6.)\ f_z(x) = f_z'(x)\\
&7.)\ \Delta = \{\delta_1(x) |\ \delta_2(x)|\ \dots |\delta_i(x)\}, \qquad \forall i: P_i \in U'_h
\end{aligned}$$

At the end of this phase, P_z broadcasts $\langle \Delta \rangle$ to all members of U.

2.2.3 Phase 3

Each member P_i receives the broadcasts and computes:

$$\begin{aligned}
&1.)\ u_{j,i} = u_{j,i} + \delta_j(i) \ mod\ q, \qquad \forall j: P_j \in U_h'\\
&2.)\ a_i = \sum_{j:P_j \in U \setminus P_r} u_{j,i} \ mod\ q
\end{aligned}$$

Similar to the adding, each member (except P_r) is now in possession of $h+1$ new member shares and the updated secret shares of the polynomial

$$f(x) = \sum_{i:P_i \in U \setminus P_r} f_i(x) \ mod\ p$$

The secret share a_r is no longer valid, hence P_r is not any longer able to cooperate with members $P_i \in U$ to reconstruct the secret a_0 using the Lagrangian interpolation. Finally each P_i deletes P_r's entry $u_{r,1}$ from its local database, removing the last trace of the membership.

3. Security

First we consider a passive adversary for our scheme. We will show that our scheme is secure against adversaries who are capable of eavesdropping on all communications channels. Later on we point out what is necessary to make the scheme secure against active adversaries.

3.1 Passive Adversary

In the distributed key generation phase, each party P_i generates a Shamir Secret Sharing scheme with a personal random polynomial $f_i(x)$. The associated shares $u_{i,j} = f_i(j)$, $\forall j \in U$ are confidentially broadcasted to all other parties. Confidentiality can be achieved by using the opposite's public key in a public key encryption scheme (e.g. El Gamal).

The security of this phase relies on the security of the used public key encryption scheme and the security of Shamir's Secret Sharing scheme, which is known as a secure k-resilient (k, n) threshold scheme. Due to the confidential transmission of the particular shares, neither an internal nor external passive adversary gains access to $k + 1$ shares that would be necessary to compromise our scheme (according to the k-resiliency of Shamir's Secret Sharing scheme).

Since our scheme is mainly based on a repeated usage of Shamir's scheme (which is formally proven secure in Appendix 1), we go without a formal security proof for our total scheme. In the following we prove the particular phases for adding and removing with informal arguments.

3.1.1 Adding: Phase 1a

With the knowledge of $h - 1 = k$ $share_i$ values, an adversary gains no advantage, because the adversary can neither recover $u_{i,0}$, nor the resulting $u_{n+1,0} = \sum share_i \bmod q$. In the case of the new $u_{i,0}$ values, the former state is unknown (only the difference $share_i$ is known) and in the case of $u_{n+1,0}$ one fully random value $share_i$ is missing. The differences δ_i give no knowledge about the former state of the secret shares $u_{i,j}$ to which they will be applied. Thus, storing old shares by an inside adversary is futile.

3.1.2 Adding: Phase 1b

The first user adds a random value r to the secret products $u_{j,1} \cdot L(n + 1, j)$. A subsequent user (w.l.o.g.) P_2 is prevented from recovering the $u_{j,1}$ values due to the addition of the random value r, which is transmitted confidentially to P_{n+1}. When P_2 knows any of the $u_{j,1}$ values, she can recover the value r and with r all other $u_{j,1}$ values. Since the ω_i are created $\forall j \in U \setminus U_h$ and user P_2 is member of U_h (by definition), she does not get knowledge of any value from $u_{j,1}$ $(j \in U \setminus U_h)$. Additionally to this method, the messages are transmitted confidentially (unicast encryption) to prevent a former user to eavesdrop the changes of the following user and hereby recovering her secrets.

3.1.2 Adding: Phase 2

To proof the security of the remaining scheme, it is necessary to show that P_{n+1} can not gain any other information from $\omega_i - r$ than $u_{i,n+1}$. ω_i is a sum of $k+1$ products of a secret ($u_{j,i}$) and a known value $L(n+1, j)$. It is not possible to disassemble this sum into the separate addends, since all possibilities have the same probability. The secret values $u_{n+1,i}$ will be sent encrypted to each user P_i, so only the recipient gets the knowledge of these values.

3.1.3 Adding: Phase 3

If some of these values are faked or compromised, the scheme will not work. In this case, all changes since the last working state (probably since the last member adding) can be undone.

3.1.4 Removing: Phase 1

The proof for the security of the removing is quite similar to the adding of members. Equal techniques are applied, so that we will only give proofs, where differences exist. $k+1$ members work together to recover the secret $u_{r,0}$ of user P_r, who should be removed from the group. This is done by computing $\omega_i \coloneqq L(0, i) \cdot u_{r,i}$, which can be sent in cleartext, since there is no need to keep the shares $u_{r,*}$ of user P_r secret. Furthermore it is possible to send difference polynomials $\delta_i(x)$ instead of the difference values for each user in U. Each user can recover$u_{r,0}$, but this will not impact the scheme negatively, hence there is no need so save the privacy of $u_{r,0}$ (or $f_r(x)$).

3.1.5 Removing: Phase 2

It is necessary to prohibit P_z from altering the $\delta_i(x)$ in such a way that the sum stays equal but the individual values differ. This can be achieved by signing $\delta_i(x)$.

3.2 Active Adversary

Considering an active adversary that is capable of manipulating sent messages and participating in the protocol. Without the knowledge of the used keys, which are used to confidentially send the data, an active adversary gains no information about the data. Firstly, wrong encrypted or random data is dropped by the participating members. Secondly, other members will not decrypt data for other parties and therefore act as a decryption oracle.

To prevent secret share manipulation, verifiable secret sharing (VSS) techniques can be used to proof the shares' authenticity [9].

5. Conclusion

This is the first time a threshold cryptosystem with the ability to remove members without changing the public key is proposed. Furthermore, our scheme does not depend on a trusted third party.

For adding and removing members, we extended the ideas from Herzberg et al. [15] that were used to refresh member shares. Our cryptographic contribution is a threshold cryptosystem that exceeds trivial extensions (for adding and removing members) of Herzberg's scheme in efficiency aspects.

Due to its dynamics, our scheme can be deployed in emerging technologies such as mesh or ad-hoc networks, where members join and leave frequently. Without a TTP, our scheme fits to all kinds of decentralized networks where the location is not fixed to a specific area. The common application of our scheme will clearly be the threshold signature (based on DL - in contrast to [1]). Besides that, our scheme is also usable in shared decryption scenarios.

When using a threshold signature with our scheme, the security property non-repudiation is provided for the group. All group members have the same functionality and equal knowledge, so that there is in fact no distinguished member. No single entity (e.g. TTP) has knowledge of the groups secret key. The group cannot deny that at least $k+1$ members worked together to compute the signature. Therefore an outside entity can be sure that a valid signature has been created by the group.

The reduction of the complexity of dynamic threshold schemes and providing an efficient non-interactive DL-based threshold signature scheme optimized for our scheme is an open challenge.

References

[1] Rosario Gennaro, Shai Halevi, Hugo Krawczyk, and Tal Rabin. Threshold rsa for dynamic and ad-hoc groups. Cryptology ePrint Archive, Report 2008/045, 2008. http://eprint.iacr.org/.

[2] Adi Shamir. Identity-based cryptosystems and signature schemes. In Proceedings of CRYPTO 84 on Advances in cryptology, pages 47-53, New York, NY, USA, 1985.

[3] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. Lecture Notes in Computer Science, 2139:213-229, 2001.

[4] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Proceedings of the 8th IMA International Conference on Cryptography and Coding, pages 360-363, London, UK, 2001.

[5] Benoît Libert and Jean-Jacques Quisquater. Efficient revocation and threshold pairing based cryptosystems. In PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing, pages 163-171, New York, NY, USA, 2003.

[6] Dan Boneh, Xuhua Ding, Gene Tsudik, and Chi Ming Wong. A method for fast revocation of public key certificates and security capabilities. In SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium, pages 22-22, Berkeley, CA, USA, 2001.

[7] Adi Shamir. How to share a secret. Communications ACM, 22(11):612-613, 1979.

[8] Yvo G. Desmedt and Yair Frankel. Threshold cryptosystems. In CRYPTO '89: Proceedings on Advances in cryptology, pages 307-315, New York, NY, USA, 1989.

[9] Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In EUROCRYPT, pages 522-526, 1991.

[10] Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi. Efficient node admission for short-lived mobile ad hoc networks. In ICNP '05: Proceedings of the 13TH IEEE International Conference on Network Protocols, pages 269-278, Washington, DC, USA, 2005.

[11] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. Lecture Notes in Computer Science, 1592:295+, 1999.

[12] Yvo Desmedt and Yair Frankel. Shared generation of authenticators and signatures (extended abstract). In CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology, pages 457-469, London, UK, 1992.

[13] Rosario Gennaro, Tal Rabin, Stanislav Jarecki, and Hugo Krawczyk. Robust and efficient sharing of RSA functions. Journal of Cryptology: the Journal of the International Association for Cryptologic Research, 13(2):273-300, 2000.

[14] Victor Shoup. Practical threshold signatures. Lecture Notes in Computer Science, 1807:207-220, 2000.

[15] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. Lecture Notes in Computer Science, 963:339-352, 1995.

Appendix

Appendix 1. Security Proof for Shamir's Secret Sharing Scheme

The scheme is secure, if it is not possible to recover the polynomial $f(x)$ (or at least the last coefficient from $f(x)$, the secret group key a_0) with less than $k+1$ shares a_i $(i>0)$. The degree of $f(x)$ is k, $|f(i)|=2^l$. We will show that an attacker owning k shares $\langle i,a_i\rangle,(i>0)$ is not successful in recovering the secret a_0.

Theorem 1:

For $k+1$ shares $\langle i,a_i\rangle,(i>0)$ with pairwise different i, there is exactly one polynomial $f(x)$ of degree $\le k$ with $f(i)=a_i,\forall i\in\{1,\dots,k+1\}$ w.l.o.g.

Proof:

A proof by contradiction states that there are two different polynomials $g(x)$ and $h(x)$ with $\deg(g)\le k$ and $\deg(h)\le k$, that can be created with $k+1$ shares $\langle i,a_i\rangle,(i>0)$. The difference polynomial $\Delta(x)=g(x)-h(x)$ must therefore have at least $k+1$ zero points. W.l.o.g. $\Delta(i)=0,\forall i\in\{1,\dots,k+1\}$.

Hence $\Delta(x)$ has a degree of maximal k, since $g(x)$ and $h(x)$ are of degree $\le k$, $\Delta(x)$ can have a maximum of k zero points. Because $\Delta(x)$ must have at least $k+1$ zero points according to our assumption (but must have only k zero points due to degree $\le k$), $\Delta(x)$ must be the null polynomial. That means, that $g(x)=h(x)$.

An attacker who only owns k shares $\langle i,a_i\rangle$ from $f(x)$ can interpolate a polynomial $p(x)$ with a maximum degree of $k-1$ (Lagrangian interpolation). If the polynomial $p(x)$ approximates $f(x)$ optimally, the polynomial has the following property w.l.o.g.:

$$f(i)=p(i),\qquad \forall i\in\{1,\dots,k+1\}$$

But additionally it must hold with a probability of $Pr\approx 1-\frac{1}{2^l}$ for each i, that:

$$f(i)\neq p(i),\qquad \forall i\notin\{1,\dots,k+1\}$$

This is because $f(x)\neq p(x)$, which follows from the fact that $f(x)$ has degree k and $p(x)$ can have a maximal degree of $k-1$. If the coefficients from $f(x)$ over $\mathbb{Z}_p^*$ $(p\in\mathbb{P})$ are uniformly distributed, the probability for $f(i)=p(i)$, $i\notin\{1,\dots,k+1\}$, is equal to the probability that a distinct value v with $f(i)=v$ is hit. The probability is

$$\Pr\left[f(i)=v,\forall i\in\mathbb{Z}_p^*\right]\approx\frac{1}{2^l}$$

We finally state the adversary's advantage of breaking Shamir's Secret Sharing scheme with only k shares is $\frac{1}{2^l}$, which is negligible for big groups.