Pseudonym Signatures: New paradigm for group communication security - Anonymity outside the group, not inside

Mark Manulis
HGI, Ruhr-University of Bochum, Germany
mark.manulis@nds.rub.de

Ahmad-Reza Sadeghi
HGI, Ruhr-University of Bochum, Germany
sadeghi@crypto.rub.de

Jörg Schwenk
HGI, Ruhr-University of Bochum, Germany
joerg.schwenk@nds.rub.de

ABSTRACT

Over the past years, much research has been done in the context of group-oriented digital signatures for group communication scenarios resulting in various kinds of signature schemes, including classical group signatures ([4], [3]) ring signatures ([20], [7]), threshold signatures ([21], [12]), multi-signatures ([14]) and traceable signatures ([15]). Regardless of this significant variety not all imaginable group communication scenarios as shown in this paper have been covered.

We describe some group communication scenarios that cannot be practically handled by currently existing schemes, and introduce a new security paradigm, called *pseudonym signatures* that can be applied in many relevant scenarios. Pseudonym signatures are related to group, ring and traceable signatures, but have some significant differences on the trust relationship between group members and their anonymity. In our scenarios all group members have equal rights and there is no group controller or any third trusted parties with extended rights. Group members are trusted not to reveal group secrets to non-members, and every group member should be able to verify computation steps concerning these secrets. The main task of pseudonym signatures is to allow in such scenarios anonymous communication between group members and non-members, and non-anonymous communication between group members at the same time. In other words, a signed message that is produced and sent by a group member reveals the signer's identity only to other group members, whereas non-members are able to verify the signature and communicate to the signer, however, without being able to identify him. Since non-members may receive signed messages from different group members they should be able to distinguish signatures of the same author. Therefore, every signature should contain a recognition parameter of its signer so that non-members can distinguish between the signer and reply to them accordingly.

We describe protocols and algorithms of pseudonym signatures and specify security requirements that must be fulfilled. We also demonstrate, how pseudonym signatures can be realized based on the existing cryptographic building blocks.

1. INTRODUCTION

Suppose in a group communication scenario group members (insiders) have to sign messages on behalf of the whole group and send them over a broadcast channel. Messages that are sent over this channel are received by all group members and also by non-members (outsiders) that are participating in the communication. Both, insiders and outsiders must be able to verify received signatures. Additionally, insiders must be able to reveal the identity of the signer from his signature, whereas outsiders are allowed only to compute the pseudonym (recognition parameter) of the signer, but not his identity. This is essential, because application may require that outsiders should be able to reply on the signed message to its author. Outsiders must also be convinced by the signature that the signer is a valid group member.

In our group communication scenarios all group members have equal rights (for example, ad-hoc communication). In such groups there is no central authority with extended rights, like group manager or third trusted party that controls the group (i.e., computes group secrets on behalf of the group). Therefore, all group members have to cooperate and control the group together. This can be achieved by the techniques of *contributory computations* (i.e., all group secrets are computed from contributions of all group members). In this paper we propose a new kind of signatures for groups that we call *pseudonym signatures*. In these signatures pseudonyms are used as recognition parameters for the authors of signed messages. Figure 1 provides a visualization of the communication scenario where pseudonym signatures may be applied. Communication participants are group members (denoted M_i) and non-members (denoted N_i). Participants who are group members cooperate to form the group, whereas non-members are participants that either are not allowed to become group members due

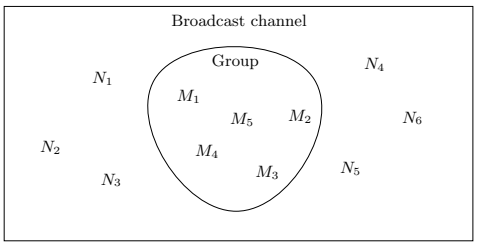

Figure 1: Group communication with pseudonym signatures

to some policy or simply do not wish to cooperate. All participants are connected to a broadcast channel that allows any participant to send a message that is then delivered to all other participants connected to the channel. The idea of pseudonym signatures is to:

- allow any group member M_i to sign messages on behalf of the group and send them over the broadcast channel such that any other group member M_j or non-member N_j can verify them. M_j should be able to identify M_i from his signatures, whereas N_j must only be convinced that M_i is a group member, without revealing his identity.
- provide to non-members pseudonyms of all group members so that any N_j can figure out, which pseudonym belongs to the author of the signed message. This allows N_j to distinguish between messages of different group members and reply to the signer of the message addressing him using this pseudonym.

Additionaly, in combination with secure symmetric and asymmetric encryption schemes pseudonym signatures

- allow group members to communicate securely over the broadcast channel using a secure symmetric encryption scheme. Any group member M_i can send encrypted messages that are received by all participants, but can be decrypted only by valid group members.
- allow any non-member N_i to send encrypted messages to the whole group over the broadcast channel using a secure asymmetric encryption scheme. Although all participants receive the encrypted message, only group members are able to decrypt it.
- allow any group member M_i to set up an authentic and secure virtual point-to-point channel on top of the broadcast channel with any non-member N_j using a secure asymmetric encryption scheme. Messages exchanged between M_i and N_j cannot be decrypted by any other participant. This secure channel preserves the anonymity of M_i since he signs messages using the pseudonym signature scheme.

We remark, that our construction incorporates key management for symmetric and asymmetric encryption during the group formation process. In the following we describe some examples, where pseudonym signatures can be applied.

Distributed Group Control. Consider a scenario of the distributed control, for example, over a battlefield operation. Given set $\mathcal{F}$ of field units and set $\mathcal{C}$ of control units, such that field units are involved in the battlefield operation and are currently controlled by some control unit $c_i \in \mathcal{C}$. In case that c_i is barred from being able to send orders (e.g., is destroyed or has lost communication), another control unit c_j should be able to take over the commandership immediately. For strategic reasons it might be desired that field units do not know, which control unit has currently a commandership, but field units must be able to verify that received orders come from an authentic control unit and also to report on the status of the operation to the current control unit. Other control units must be prepared to take over the commandership at any time, therefore, they have to be able to log the orders of c_i and reports from field units. Pseudonym signatures can be used in this scenario as shown in Figure 2. All control units from $\mathcal{C}$ form a group.

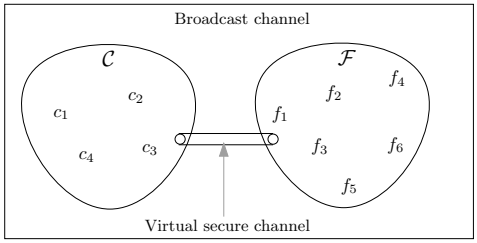

Figure 2: Pseudonym signatures in a battlefield

Current control unit c_i broadcasts its signed encrypted orders so that every other control unit and every field unit receive them and are able to verify that they come from an authentic control unit. Since pseudonym signatures do not reveal the identity of the signer, field units cannot determine who c_i is, but other control units from the group are able to do this. Since field units can distinguish between signatures of the same signer, they are able to report on the status of operation to c_i. Other control units receive these reports too and can follow the development of the operation.

Consultative Group Decision. Consider a group of people $\mathcal{G}$ who wish to make their common decision based on anonymous consultations with third parties in $\mathcal{T}$. Every member of $\mathcal{G}$ is allowed to initiate an anonymous consultation with any $t_i \in \mathcal{T}$ such that only other group members can identify the initiator and follow the consultation process. Third party t_i might be queried from different members at the same time and must, therefore, be able to determine to whom it should respond on each query. Examples for such group decision making scenarios are: anonymous review and discussion of submitted papers by the program committee, or selection of participants for a project tender based on submitted proposals. Group members use pseudonym signatures to sign their messages to t_i. For example, if a reviewer wishes to discuss several topics of the submitted paper with its author without revealing own identity, he can use pseudonym signature scheme to sign his messages to the author. The author is able to distinguish between queries of different reviewers using signer's

pseudonym and respond accordingly.

Anonymous Intergroup Communication. Pseudonym signature schemes allow authentic anonymous communication between members of multiple groups. Members of different groups can exchange signed messages without revealing own identities to members of other groups. Only members of the own group are able to identify them from their signatures. For example, two teams want to carry out a fair two-player game competition (Figure 3). Assume

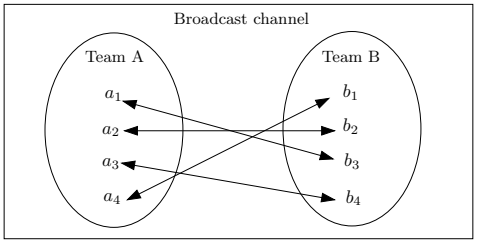

Figure 3: Pseudonym signatures in a two-player game competition

that players of one team know strengths and weaknesses of players of another team. Thus, if a player knows its opponent then he might benefit from this knowledge during the game. Therefore, to achieve fairness it is preferable that players do not know their opponents. However, once the game is started, both players have to be able to verify that all game moves come from the same opponent. In Figure 3 virtual channels between each two opponents are represented by arrows. After competition is finished, both teams should be able to figure out the authentic result, and can also reveal identities of both opponent players of each pair.

We argue in Section 4 that existing signature schemes for groups, like classical group signatures, traceable signatures, and ring signatures cannot be applied to the described group communication scenarios.

1.1 Conventions

In this section we introduce basic notations and definitions used throughout this paper.

1.1.1 Roles

The communication participants who actively cooperate on the group formation are called *group members* or *insiders*. A group member who generates a pseudonym signature on a message is called a *signer*. The communication participants who are not involved in the group formation process are called *non-members* or *outsiders*. The communication participant who verifies a pseudonym signature of a message is called *verifier*. Both, insiders and outsiders are able to take verifier's role.

1.1.2 Cryptographic primitives

An encryption scheme is denoted with the tuple $(KGen(), Enc(), Dec())$ for key generation, encryption and decryption algorithms. In case of symmetric encryption the shared secret key is denoted k. A digital signature scheme is denoted with the tuple $(KGen(), Sig(), Ver())$ for key generation, signing and verification algorithms. The tuple (y_i, x_i) denotes the public and the private key of participant i, respectively. Sometimes we denote public keys with pk and private keys with sk.

2. MODEL

2.1 Protocols and Algorithms

In this section we describe protocols and algorithms of a pseudonym signature scheme.

Definition 1. A *pseudonym signature scheme* $\mathcal{PSS} = \{Setup(), PSign(), PVerify(), PTrace()\}$ is a digital signature scheme that consists of:

- An interactive randomized protocol $Setup()$ between all members actively involved in the group formation that on input a number of members $n \in \mathbb{N}$ outputs a set of members' *identities ID*, a set of members' *pseudonyms PS*, and a set of members' *secret signing keys SK*. Values id_i, ps_i, and sk_i denote the identity, the pseudonym, and the secret signing key of a member, respectively. sk_i is known only to member with id_i.
- A randomized algorithm $PSign()$ that on input a secret signing key $sk \in SK$ and a message m, outputs a signature σ on m.
- A deterministic algorithm $PVerify()$ that on input a candidate signature σ, a message m, and the set of pseudonyms *PS* outputs either a pseudonym ps or $\perp$ if a failure occurs.
- A deterministic algorithm $PTrace()$ that on input a candidate signature σ, a message m, any secret signing key $sk \in SK$, the set of pseudonyms *PS*, and the set of identities *ID* outputs either an identity id or the symbol $\perp$ if a failure occurs.

We call σ a *pseudonym signature*, since pseudonyms are returned by the verification algorithm.

A pseudonym signature scheme $\mathcal{PSS}$ from Definition 1 is correct if for all $n \in \mathbb{N}$, all outputs (ID, PS, SK) of $Setup()$ with $sk_i, sk_j \in SK$, $id_i \in ID$, $ps_i \in PS$, $m \in \{0,1\}^*$, and σ of $PSign(sk_i, m)$ the following holds:

$$PVerify(\sigma, m, PS) = ps_i \text{ and}$$
$$PTrace(\sigma, m, sk_j, PS, ID) = id_i$$

2.2 Security Requirements

2.2.1 Anonymity Attacks

In an anonymity attack against the pseudonym signature scheme the adversary algorithm *A* (an outsider) tries to figure out the identity id_i of the signer of a pseudonym signature σ. *A* is allowed to perform chosen message attacks, i.e., *A* obtains a set of pseudonym signatures $\mathcal{S}$ on any messages of its choice and succeeds if it is able to compute an identity id_i that corresponds to at least one pseudonym signature from $\mathcal{S}$. A pseudonym signature scheme is called *anonymous* if for any polynomial time adversary algorithm *A* the probability of computing the signer's identity is negligible.

2.2.2 Traceability Attacks

In a traceability attack against the pseudonym signature scheme the adversary algorithm *A* (an outsider, or an insider, or a collusion of insiders) is allowed to compute a set of corrupted members ID_A (i.e., *A* knows secret signing keys of all members in ID_A.), and tries to produce a forged signature that either cannot be traced to any group member or can be traced to a group member that is not corrupted by *A*. Algorithm *A* starts its attack by adaptively corrupting a set ID_A of group members and learning the set SK_A of their secret signing keys. *A* feels free to choose the identities and the number of corrupted members (at least one member should not be corrupted). At the end of the attack *A* outputs (m, σ) and succeeds if $PTrace()$ returns either $\perp$ or an identity id_i such that $id_i \notin ID_A$ and $id_i \in ID$. During its attack *A* is allowed to obtain pseudonym signatures from any group member with id_i on any message m' of its choice (i.e., chosen message attack). A pseudonym signature scheme is called *traceable* if for any polynomial time adversary algorithm *A* the probability of computing the successful forgery is negligible. It is obviously, that this traceability attack incapsulates unforgeability, exculpability, collusion and framing attacks.

2.3 Discussion on Sizes

From Definition 1 we can straightforward the lower size bounds of sets *ID*, *PS* and *SK*. Since every member has an own unique identity, a pseudonym and a secret signing key it is unavoidable that the lower size bound of all these sets is n. To the contrary, it is preferrable that the size of pseudonym signatures does not grow linearly with the number of group members, i.e., remains *constant*.

3. REALIZATION

3.1 Building Blocks

Contributory Group Key Agreement

In order to handle the control of the group secrets according to the requirement of equal rights of group members we suggest to use *contributory group key agreement* (CGKA) protocols, like [17], or [18]. This protocols allow every group member to submit own contribution and compute the group key k_G as a function of all submitted contributions. CGKA protocols fulfill the following requirements:

- *Computational group key secrecy* requires that for a passive adversary it must be computationally infeasible to discover any secret group key.
- *Decisional group key secrecy* requires that for a passive adversary it must be computationally infeasible to distinguish any bits of the secret group key from random bits.
- *Forward secrecy* requires that any passive adversary being in possession of a subset of old group keys must not be able to discover any subsequent group key.
- *Backward secrecy* requires that any passive adversary being in possession of a subset of contiguous group keys must not be able to discover any preceding group key.
- *Key independence* requires that any passive adversary being in possession of any subset of group keys must not be able to discover any other group key.

For more details and comparance of existing CGKA protocols we refer to [2].

Member's contributions in these CGKA protocols have the following form. Every member chooses own secret x_i and computes his contribution $y_i = g^{x_i}$ where g is a generator of an algebraic multiplicative prime order group $\mathbb{G}$ where the discrete logarithm assumption holds (e.g., $\mathbb{Z}_p^*$ with prime p). The discrete logarithm assumption means that given value g^x it is computationally infeasible to compute x for all $x \in \mathbb{Z}_{|\mathbb{G}|}$. Pseudonym signatures benefit from such construction, because contribution y_i can be used at the same time as a public key and secret x_i as a private key of member M_i. For simplicity, we use public key of member M_i as his identity, i.e., $id_i = y_i$.

Signatures of Knowledge

Signature of knowledge schemes, introduced in [10], are message dependent zero-knowledge proofs of knowledge of a secret s that are made non-interactive using the Fiat-Shamir heuristic [13]. These schemes consist of two algorithms ($SKSig()$, $SKVer()$), where $SKSig()$ is a randomized signing algorithm and $SKVer()$ a deterministic verifying algorithm. A signer *S* who is in possession of some secret s can compute the signature of knowledge of s on a message m (denoted $SK[s](m)$) using $SKSig()$ and send it to a verifier *V*. If algorithm $SKVer()$ performed by *V* accepts $SK[s](m)$ then *V* is convinced that *S* knows s but learns nothing about this secret. However, if *S* does not know s or does not use s in $SKSig()$ to compute $SK[s](m)$ then $SKVer()$ rejects the signature. A signature of knowledge scheme is called secure if the probability of producing a forged signature $SK'[s](m)$ without knowing the secret s such that $SKVer()$ accepts $SK'[s](m)$ is negligible, and if any correctly generated $SK[s](m)$ does not reveal any information about s to a verifier *V*.

3.2 Basic Construction

In this section we describe a basic construction of a pseudonym signature scheme for a group of members $\{M_1, \ldots, M_n\}$. We begin with the protocol $Setup()$. Every member M_i generates own contribution (considered as his public key) y_i from the chosen private key x_i as described above. The identity of member M_i in the context of the group communication session is $id_i = y_i$. Members perform the specified secure CGKA protocol and compute the secret group key k_G. Then, every member M_i generates a second key pair $(\tilde{y}_i, \tilde{x}_i)$, and uses public key $\tilde{y}_i$ as his pseudonym, thus $ps_i = \tilde{y}_i$. The secret signing key sk_i of member M_i is a tupel $(k_G, x_i, \tilde{x}_i)$. Next, group members exchange their pseudonyms without revealing them to non-members that are connected to the same broadcast channel. This can be achieved using a secure symmetric encryption scheme with the secret group key k_G. Every member should also proof the authenticity of his pseudonym by signing it using a digital signature scheme with his private key x_i. Thus, every member M_i computes the signature $s_i = Sig(x_i, (ps_i))$, encrypts it to $e_i = Enc(k_G, (ps_i, s))$ and sends e_i over the broadcast channel. Obviously, only group members are able to decrypt e_i's and verify encrypted signatures

s_i, so that only group members learn a pair (id_i, ps_i) that belongs to the member M_i. One of the requirements on pseudonym signatures is that non-members must know all pseudonyms of group members, but not their identities. For this purpose one group member, w.l.o.g. M_1, computes a random permutation of all pseudonyms *PS* and sends it together with *ID* signed with own private key x_1 (i.e., $Sig(x_1, (PS, ID))$) over the broadcast channel. Every group member is able to verify whether *ID* and *PS* sent by M_1 are complete, i.e., consists of pseudonyms ps_i and identities id_i of all group members. If a group member M_i notices that *ID* or *PS* are not complete he resends them together with $Sig(x_i, (PS, ID))$ and a proof π for incompleteness of *ID* or *PS* that have been sent by M_1. This control technique fulfills the requirement on equal rights of all group members, and guarrantees that non-members receive authentic and complete *ID* and *PS*. Thus, non-members learn pseudonyms and identities of all group members, but are not able to build corresponding pairs (id_i, ps_i) for same i's. The order of pseudonyms in *PS* is randomized to eliminate any correlations with *ID*.

During the communication session M_i can sign messages using algorithm $PSign()$ that can incapsulate any secure digital signature scheme ($KGen()$, $Sig()$, $Ver()$). M_i signs a message m using his private key $\tilde{x}_i$ from sk_i, i.e. M_i computes the pseudonym signature $\sigma_i = Sig(\tilde{x}_i, m)$. Per construction only M_i knows sk_i, thus if the digital signature scheme is secure, it is not possible for any adversary to produce a forged pseudonym signature that would be traced to M_i.

Upon receiving m and σ_i all group members and non-members are able to verify the signature using algorithm $PVerify()$ that incapsulates the digital signature scheme's verifying algorithm $Ver(\tilde{y}_i, \sigma_i, m)$, where $\tilde{y}_i = ps_i$ and $ps_i \in PS$.

Algorithm $PTrace()$ can be performed only if $PVerify()$ accepts the signature. $PTrace()$ finds a pair (id_i, ps_i) from the set of pairs computed during $Setup()$ and outputs $id_i \in ID$. Obviously, only insiders are able to perform $PTrace()$ since only they learn all pairs (id_i, ps_i) during $Setup()$. Since non-members know only pseudonyms ps_i they are not able to identify the signer of the message.

Discussion: This basic construction requires: (a) every member M_i to generate two private/public key pairs (y_i, x_i) and $(\tilde{y}_i, \tilde{x}_i)$, (b) additional communication and computation to exchange pseudonyms, including one signature generation for s_i, one encryption for e_i, $n-1$ verifications for all s_j and $n-1$ decryptions for all e_j per every group member. In the following section we propose a non-trivial improvement of this solution that requires only one private/public key pair (y_i, x_i) per member M_i, and no additional communication between group members to exchange their pseudonyms.

3.3 More Efficient Construction

The protocol $Setup()$ of the more efficient construction is related to that of the basic construction. Like in the basic construction, every insider M_i generates own public key y_i from the chosen private key x_i as $y_i = g^{x_i}$, and submits it as own contribution to the specified secure CGKA protocol to compute the secret group key k_G. In difference to the basic construction, group members compute the public group key $pk_G = g^{k_G}$ where g is a generator of $\mathbb{G}$ as described above. The identity of a group member M_i in the context of the communication session is set to $id_i = y_i$. Another difference to basic construction is that every group member computes pseudonyms of all other group members without any additional interaction. Every member M_i computes his pseudonym $ps_i = f(k_G, id_i)$ where f is a specified one-way trapdoor function with the secret group key k_G as a trapdoor. The function f fulfills the following requirements: only if the trapdoor k_G is known, then f and inverse function f^{-1} can be computed, i.e. ps_i can be computed from id_i and vice versa; otherwise, it is computationally infeasible to perform these computations. We suggest to choose f either as: (a) secure symmetric encryption scheme ($KGen()$, $Enc()$, $Dec()$), i.e. $ps_i = Enc(k_G, id_i)$ and $id_i = Dec(k_G, ps_i)$, or (b) f, f^{-1} as modular exponentiation functions such that $ps_i = id_i^{k_G}$ and $id_i = ps_i^{{k_G}^{-1}}$ where g is a generator of $\mathbb{G}$, $id_i, ps_i \in \mathbb{G}$, and $x_i \in \mathbb{Z}_{|\mathbb{G}|}$ (also note, that $id_i^{k_G} = pk_G^{x_i}$, because of $id_i = y_i = g^{x_i}$ and $pk_G = g^{k_G}$). Using f with the secret group key k_G as a trapdoor allows other group members to compute the whole pseudonym set *PS* without additional interaction with other group members. The secret signing key sk_i of member M_i is a tuple (k_G, x_i). One group member, w.l.o.g. M_1, computes a random permutation of all pseudonyms *PS* and sends it together with *ID* and pk_G signed with own private key x_1 (i.e., $Sig(x_1, (PS, ID, pk_G))$) over the broadcast channel (in difference to the basic construction pk_G is included in the message). Upon receiving *ID*, *PS* and pk_G other group members verify whether both sets are complete and the public group key pk_G is correct. As in the basic construction, a group member that notices the incompleteness of *PS* or *ID* or incorrectness of pk_G resends the message with the corresponding proof. Thus, non-members receive authentic and correct values for pk_G, *ID* and *PS* at the end of $Setup()$ and are not able to build corresponding pairs (id_i, ps_i) for same i's.

A group member M_i can produce his pseudonym signature σ_i on message m using algorithm $PSign()$ that incapsulates the signing algorithm $SKSig()$ of the secure signature of knowledge scheme $(SKSig(), SKVer())$. Algorithm $PSign()$ uses $SKSig()$ to compute the signature of knowledge of $sk_i = (k_G, x_i)$ and its relation to a pseudonym $ps_i \in PS$ on a message m as described in Section 3.1 and denoted $SK[sk_i](m)$ in the following. For example, if f is a symmetric encryption scheme ($KGen()$, $Enc()$, $Dec()$) then $SK[sk_i](m)$ shows that the signer knows x_i that was used to compute $ps_i = Enc(k_G, g^{x_i})$ without revealing the identity $id_i = g^{x_i}$, and if f is a modular exponentiation function such that $ps_i = id_i^{k_G}$ then $SK[sk_i](m)$ shows that the signer knows x_i according to $ps_i = pk_G^{x_i}$.

$PVerify()$ accepts the pseudonym signature σ on a message m if incapsulated verification algorithm $SKVer()$ accepts the signature of knowledge $SK[sk_i](m)$. In this case $PVerify()$ returns the pseudonym $ps_i \in PS$ of the signer, else it returns $\perp$. Thus, only if $PVerify()$ accepts σ non-members learn the pseudonym of its signer.

Algorithm $PTrace()$ can be performed only if $PVerify()$ accepts the pseudonym signature and returns ps_i*PS*. Algorithm $PTrace()$ reveals the identity of the signer by computing $id_i \in$ *ID* from ps_i through the computation of the inverse function f^{-1}. Obviously, only insiders are able to perform $PTrace()$ since only they learn the trapdoor k_G after performing the CGKA protocol during $Setup()$. Non-members do not know k_G. Therefore, it is computationally infeasible for them to compute f^{-1}, and so to identify the signer of the signature.

Security: Intuitively, the more efficient construction fulfills requirements of anonymity and traceability from Section 2.2. In order to break anonymity, adversary algorithm *A* has to compute the identity id_i either from the pseudonym $ps_i \in$ *PS* or from a pseudonym signature σ_i produced by member M_i using $PSign()$. Obviously, the computation of id_i from ps_i without knowing the trapdoor k_G relies on the security of the one-way trapdoor function f and on the security of the CGKA protocol. Suppose *A* is able to compute id_i from ps_i. If CGKA protocol is secure, i.e. fulfills requirements from Section 3.1, then it can be shown that function f does not fulfill security requirements of a one-way trapdoor function. However, if f is a secure one-way trapdoor function, then it is possible to show that *A* can compute k_G, thus CGKA protocol does not fulfill its security requirements. The above construction of the pseudonym signature scheme is secure if secure CGKA protocol and secure one-way trapdoor function f are used.

In order to break tracebility, *A*[1] has to produce a forged pair (m, σ) such that $PVerify()$ accepts it and returns a pseudonym $ps \in$ *PS*, and $PTrace()$ returns either $\perp$ or $id_i \notin$ *ID*$_A$[2] as described in Section 2.2.2. The probability that $PVerify()$ accepts σ and $PTrace()$ returns $\perp$ is negligible, because $PVerify()$ returnes $ps \in$ *PS* and all pseudonyms in *PS* have been already verified by group members during $Setup()$. The probability that $PVerify()$ accepts σ and $PTrace()$ returns $id_i \notin$ *ID*$_A$ depends on the security of the incapsulated signature of knowledge scheme $(SKSig(), SKVer())$. It is possible to show that if *A* computes a successful pseudonym signature forgery (m, σ) that can be traced to a group member with $id_i \in$ *ID*, then one can compute a successful forgery of the signature of knowledge $SK'[sk_i](m)$. Thus, if secure signature of knowledge scheme is used, then the above construction of the pseudonym signature scheme fulfills the traceability requirement.

4. RELATED WORK

Although there exist various digital signatures for groups, pseudonym signatures differ from them in trust relationship and signer's anonymity as shown in the following.

Group signatures introduced by Chaum *et. al.* [11] and further studied and improved in [10], [9], [4], [16], [6], [5], [8] and [3] allow members of a group to sign messages on behalf of the group so that it is not possible for a verifier to compute the signer's identity. Nevertheless, there exists a designated *group manager* that can open the group signature in the case of dispute, i.e., reveal the identity of its signer. In the context of pseudonym signatures we are interested in two security requirements: *unlinkability*, that is no party except for the group manager is able to relate two or more signatures as being produced by the same signer, and *anonymity*, that is no party except for the group manager is able to reveal the signer's identity from the signature ([4]). There are several differences between group signatures and pseudonym signatures. First, in group signatures all group members trust the group manager that computes all group secrets. This is obviously a contradiction to the requirement of equal rights of group members in pseudonym signature schemes. Secondly, group signatures are unlinkable. Thus, group members and non-members that receive a signed message are not able to relate it to any previously received signed message and figure out which messages have been signed by the same group member. The only party that is able to do this is the group manager. Obviously, this is an obstacle for the communication in scenarios described above. Third difference is given by the anonymity requirement of group signatures that allows no participant, except for the group manager, to reveal the identity of the signer from the group signature. In pseudonym signatures, however, all group members must be able to do this. To overcome this obstacle the group manager must open every signature and send the identity of its signer together with an adequate proof securely to other group members. More specific, let σ_i be a group signature of the group member with identity id_i. The group manager is the only party that is able to compute id_i from σ_i. Thus, group manager computes id_i for every gerenerated signature σ_i, encrypts the identity and the signature as $Enc(k_G, (id_i, \sigma_i))$, where k_G is the secret group key chosen by the group manager during setup and distributed to all group members during their group registration procedure, and sends this value over the broadcast channel. This would ensure that only group members can decrypt corresponding pairs (id_i, σ_i). However, this solution is less practicable due to the increasing communication complexity and interaction during the signature generation (i.e., computations and additional message of the group manager can be considered as such interaction), and due to the fact that the group manager becomes a single-point of failure and attack.

Therefore, group signature schemes cannot be applied to the communication scenarios that are specified for pseudonym signature schemes. Even if the group manager would open every group signature and send the identity of its signer together with an adequate proof securely to other group members, the obstacle of the centralized trust relationship and the problem with signers' recognition parameters cannot be solved without changing the definition of group signatures.

Traceable signatures introduced by Kiayias *et. al.* [15] extend group signatures as follows. There exist several group members, called *tracers* that receive so-called *tracing trapdoors* from the group manager and find all group signatures that correspond to this tracing trapdoor. The tracing trapdoor reveals no information about the identity of the corresponding group member. Thus, tracers can be used to collect all group signatures of a certain group member

[1] *A* is an outsider, an insider or a collusion of group members

[2] *ID*$_A$ is the set of group members' identities corrupted by *A*, i.e., *A* knows sk_i of any M_i with $id_i \in$ *ID*$_A$

without being able to identify him, while the group signatures produced by other group members remain unlinkable and cannot be collected, unless the group manager reveals their tracing trapdoors too. Like in classical group signatures the group manager is the only participant who is able to identify the signer of a signed message. Another property of traceable signatures is that group members are able to produce claiming proofs for their group signatures, i.e., a group member can convince any verifier that he is the author of a certain signed message.

In the context of pseudonym signatures tracing trapdoors can be seen as pseudonyms that allow tracers to follow the communication by linking signed messages of the same authors. Since, by definition, tracing trapdoors are only known to the group manager that reveals them to the tracers, other group members and also non-members are not able to recognize whether several signed messages have been produced by the same author. The property of traceable signatures that allows group members to claim their signatures can be utilized to overcome this obstacle. The signer generates a signature and its claiming proof. The signature is sent to all communication participants including group members and non-members, whereas the claiming proof is sent only to group members that are then able to identify the signer by verifying the proof. This can be achieved by encrypting the proof with the secret key of the group. More specific, suppose the signer with id_i generates a signature σ_i and its claiming proof π. The proof π shows that the author of the signature σ_i on message m has the identity id_i. The signer encrypts proof π, i.e., computes $\Pi = Enc(k_G, \pi)$ where k_G is the secret group key chosen by the group manager and distributed to other group members during their registration. The signer also attaches Π to σ_i before sending the signed message over the broadcast channel. Outsiders would be able to verify σ_i, and insiders would additionally be able to decrypt $\pi = Dec(k_G, \Pi)$ and verify it. If verification of π is successful, then the signer's identity can be revealed. Using encryption scheme is required to prevent outsiders from obtaining π and learning the signer's identity. Obviously, the drawback of this realization is increased communication complexity, as well as increased computation complexity due to additional generation of claiming proofs and their encryption. To overcome the second problem with anonymity revokation inside the group, the group manager must open every produced signature and send the identity of its signer together with an adequate proof securely to other group members as described in the context of group signatures above. However, the third problem concerning the trust relationship between group members, i.e, equal rights for all group members remains unsolved, like in case of classical group signatures.

Ring signatures introduced by Rivest *et. al.* [20] and further studied in [1], [7], [22] and [19] do not require any group manager or any cooperation of members to form the group. For the signature generation every participant may choose the group of possible signers (i.e., set of their public keys) including himself and produce signed messages that convince a verifier that the author belongs to this group. In ring signatures users might even not know about their membership in a certain group. This is infeasible to achieve with classical group signature schemes where signers are only registered group members. Ring signatures provide unconditional anonymity. Thus, neither other group members nor any non-members are able to reveal the signer's identity (i.e., his public key) from the ring signature.

Pseudonym signatures and ring signatures have in common that there is no central authority, like the group manager who controls the group. However, all group members in pseudonym signature schemes have to cooperate during the protocol $Setup()$, whereas in ring signatures any participant can specify own set of possible signers without cooperation with other communication participants. Another common property of pseudonym signatures and ring signatures is that a non-member that shares the broadcast channel with the group members may learn the possible set of signers (i.e., their identities), but is not able to reveal the identity of the signer from the signature. However, pseudonym signatures allow other group members to reveal this identity, whereas in ring signatures even other members are not allowed to do this. The third difference is that ring signatures do not provide any recognition parameters for the signers. Thus, communication participants are not able to distinguish between signed messages. This makes continuous communication as described in our scenarios impossible.

Obviously, some of existing variations of signature schemes for group communication are related to pseudonym signatures, however, none of them fulfills all requirements of pseudonym signatures. Thus, none of existing schemes can be used instead of pseudonym signatures in the described scenarios, unless its definition and requirements are changed. Therefore, we consider pseudonym signatures as a new paradigm for securing various group communication scenarios covering a wide range of applications.

5. REFERENCES

[1] M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In *ASIACRYPT '02: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2501 of *Lecture Notes in Computer Science*, pages 415–432. Springer-Verlag, 2002.

[2] Y. Amir, Y. Kim, C. Nita-Rotaru, and G. Tsudik. On the performance of group key agreement protocols. *ACM Trans. Inf. Syst. Secur.*, 7(3):457–488, 2004.

[3] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO '00: Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer-Verlag, 2000.

[4] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology (EUROCRYPT 2003),Lecture Notes in Computer Science*, volume 2656, pages 614–629. Springer-Verlag Berlin, 2003.

[5] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA*, pages 136–153, 2005.

[6] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Berlin: Springer-Verlag, 2004.

[7] E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. In *Advances in Cryptology - CRYPTO 2002: 22nd Annual International Cryptology Conference*, volume 2442 of *Lecture Notes in Computer Science*, pages 465–480. Springer-Verlag, 2002.

[8] J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. In *SCN*, pages 120–133, 2004.

[9] J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In *Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security*, pages 160–174. Springer-Verlag Berlin, 1998.

[10] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology, Lecture Notes in Computer Science*, volume 1294, pages 410–424. Springer-Verlag, 1997.

[11] D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology (EUROCRYPT 1991),Lecture Notes in Computer Science*, volume 547, pages 257–265. Springer-Verlag Berlin, 1991.

[12] I. Damgård and K. Dupont. Efficient threshold rsa signatures with general moduli and no extra assumptions. In *Public Key Cryptography - PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography*, pages 346–361, 2005.

[13] A. Fiat and A. Shamir. How to prove yourself: Practical solution to identification and signature problems. In *Advances in Cryptology - CRYPTO 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987.

[14] P. Horster, M. Michels, and H. Petersen. Meta-multisignature schemes based on the discrete logarithm problem. In *Proc. of IFIP/SEC'95*, pages 128–141. Chapman & Hall, 1995.

[15] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589. Springer-Verlag, 2004.

[16] A. Kiayias and M. Yung. Extracting group signatures from traitor tracing schemes. In *Advances in Cryptology (EUROCRYPT 2003),Lecture Notes in Computer Science*, volume 2656, pages 630–648. Springer-Verlag Berlin, 2003.

[17] Y. Kim, A. Perrig, and G. Tsudik. Communication-efficient group key agreement. In *Information Systems Security, Proc. of the 17th International Information Security Conference, IFIP SEC'01*, 2001.

[18] Y. Kim, A. Perrig, and G. Tsudik. Tree-based group key agreement. *ACM Transactions on Information and System Security*, 7(1):60–96, 2004.

[19] J. K. Liu, V. K. Wei, and D. S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In *Information Security and Privacy: 9th Australasian Conference, ACISP 2004.*, volume 3108 of *Lecture Notes in Computer Science*, pages 325–335. Springer-Verlag, 2004.

[20] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer-Verlag, 2001.

[21] V. K. Wei. A bilinear spontaneous anonymous threshold signature for ad hoc groups. Cryptology ePrint Archive, Report 2004/039, 2004. http://eprint.iacr.org/.

[22] C. zhi Gao, Z. an Yao, and L. Li. A ring signature scheme based on the Nyberg-Rueppel signature scheme. In *Applied Cryptography and Network Security, First International Conference, ACNS 2003.*, volume 2846 of *Lecture Notes in Computer Science*, pages 169–175. Springer-Verlag, 2003.