Separability and Efficiency for Generic Group Signature Schemes

(Extended Abstract)

Jan Camenisch*
BRICS**
Department of Computer Science
University of Aarhus
Ny Munkegade
DK – 8000 Århus C, Denmark
camenisch@ieee.org

Markus Michels
Entrust Technologies Europe
r3 security engineering ag
Glatt Tower, 6th floor
CH – 8301 Glattzentrum,
Switzerland
Markus.Michels@entrust.com

Abstract. A cryptographic protocol possesses separability if the participants can choose their keys independently of each other. This is advantageous from a key-management as well as from a security point of view. This paper focuses on separability in group signature schemes. Such schemes allow a group member to sign messages anonymously on the group's behalf. However, in case of this anonymity's misuse, a trustee can reveal the originator of a signature. We provide a generic fully separable group signature scheme and present an efficient instantiation thereof. The scheme is suited for large groups; the size of the group's public key and the length of signatures do not depend on the number of group member. Its efficiency is comparable to the most efficient schemes that do not offer separability and is an order of magnitude more efficient than a previous scheme that provides partial separability. As a side result, we provide efficient proofs of the equality of two discrete logarithms from *different* groups and, more general, of the validity of polynomial relations in $\mathbb{Z}$ among discrete logarithms from *different* groups.

1 Introduction

Multiparty cryptographic protocols are typically preceded by a setup phase in which the involved entities choose their public and secret keys besides other system-parameters. Often their choices depend on the other entities' choices. Not only forces this the entities to choose new key-pairs for each (kind of) protocol they participate in but also to re-choose their keys if other entities renew theirs or if the system-parameters are changed. Apart from such key-management problems, a dependency between different keys can cause security problems as well. For instance, if some entity choose weak parameters or discloses his/her secret keys, other entities' security can be weakened as well.

* Current address: IBM Zurich, Säumerstrasse 4, CH-8803 Rüschlikon.

** Basic Research in Computer Science, Center of the Danish National Research Foundation.

Michael Wiener (Ed.): CRYPTO'99, LNCS 1666, pp. 413–430, 1999.

To overcome such problems the concept of *separability* introduced in [30] is promoted and refined. A cryptographic protocol is said to enjoy *perfect* separability if all entities can chose not only their keys independently of the other entities but also any instance of the required cryptographic primitives, e.g., any existentially unforgeable signature scheme. A protocol has *strong* separability if the entities are restricted in their choice of the instances of the primitives, e.g., only the RSA [43] signature scheme with a 1024 bit modulus, and it possesses *weak* separability if their choices need to depend on some system parameters, e.g., the generator of some prime order algebraic group. Hence, weak separability does not overcome potential key-management problems but is satisfactory from a security point of view. Perfect, and to some extent also strong separability, is a prerequisite that a cryptographic scheme can be set-up in absence of all potentially involved parties: it suffices that their public keys are authentically available (and satisfy certain requirements in case of strong separability). Furthermore, a cryptographic protocol can only have *partial* separability of any kind, that is, separability only with respect to some of the participating entities. At the cost of prohibitive inefficiency perfect separable protocols can in most cases be obtained by combining different cryptographic primitives with general, (e.g., circuit-based) zero-knowledge proof techniques.

In this paper we consider group signature schemes and show how perfect separability can be achieved by providing a generic but potentially inefficient solution. We then restrict ourselves to strong separability and point out how efficiency can be gained. A group signature scheme allows group-members to sign messages anonymously and unlinkably on behalf of the group. To counterfeit misuse of this anonymity the scheme enables a third party, called revocation manager, to reveal the identity of a signature's originator. Since its introduction by Chaum and van Heyst [17], a number of researchers have proposed more efficient solutions and diversified the model. The first schemes presented [8,17,18,40] have the property that the length of signatures and/or the size of the group's public key depend on the size of the group and thus they are not suited for large groups. This drawback is overcome by the schemes presented in [9,11] as well as by a further one in [12][1]. The less efficient scheme of [11] was shown to be insecure but can easily be adjusted [2]. However, none of these schemes enjoys separability since the revocation manager's keys depend on the ones chosen by the membership manager.

A concept dual to group signature schemes is identity escrow [30]. It can be regarded as a group identification scheme with revocable anonymity. In fact, any identity escrow scheme with a 3-move identification protocol can be turned into a group signature scheme by applying the Fiat-Shamir heuristic [23] to the identification protocol; the opposite is achieved by signing a random message and then proving the knowledge of a signature on the chosen message. The schemes

[1] Other schemes having the same properties are either completely broken [31,38] as shown in [33,34] or do not satisfy the usual security requirements (i.e., the scheme proposed in [1] is not secure against colluding group members while in the protocol given in [32] signatures are linkable).

presented in [30] are partially separable (according to our nomenclature), i.e., strongly separable with respect to the revocation manager but not with respect to group members. Due to their construction, they are suitable for large groups as well but are much less efficient compared to the schemes presented in [9,11,12]. The schemes presented in this paper possess strong separability with respect to all participants and their efficiency is in the same order of magnitude as the ones in [9,11,12].

As a side result, we present efficient proofs of the validity of polynomial relations in $\mathbb{Z}$ among discrete logarithms from *different* groups, which might be of independent interest, in particular, for obtaining separable schemes for other cryptographic scenarios.

2 A Model of Separable Group Signature Schemes

This section describes the model of separable group signature schemes and states the security requirements. The main difference to the model of ordinary group signature schemes is that here the key generation of the membership manager, the revocation manager, and the group members are individual procedures that are independent of each other. We assume only a single revocation manager and membership manager, but the definition extends easily to several of them.

Definition 1. *Let ℓ_M, ℓ_R, and ℓ_U be security parameters. A separable group signature scheme consists of the following procedures:*

GKG-MM: *A probabilistic algorithm that on input 1^{ℓ_M} outputs the membership manager's secret key x_M and public key y_M.*

GKG-RM: *A probabilistic algorithm that on input 1^{ℓ_R} outputs revocation manager's secret key x_R and public key y_R.*

GKG-GM: *A probabilistic algorithm that on input 1^{ℓ_U} outputs a group member's secret key x_U and public key y_U.*

GKG-S: *A probabilistic algorithm that generates the system parameter y_S and an empty group member list GML. This list is a public board which anybody can read but only the membership manager and a potential group member U together can add entries related to U's identity.*

Reg: *A probabilistic interactive protocol between the group member U and the membership manager. Their common input is the group member's identity ID_U and public key y_U. If both parties accept, their common output is the membership certificate s_U on y_U. Finally, the two parties add y_U and ID_U to the (public) group member list GML.*

GSig: *A probabilistic algorithm that on input s_U, x_U, the group's public key Y, and a message m outputs a group signature s on m.*

GVer: *An algorithm that on input the group's public key Y, an alleged signature s, and a message m outputs 1 if and only if the signature is valid w.r.t. Y.*

GTrace: *An algorithm which on input the revocation manager's secret key x_R, the group's public key Y, a message m, and a signature s on m outputs the identity*

ID_U of the originator of the signature and a proof V that ID_U is indeed the originator.

The group's public key consists of the triple $Y = (y_R, y_M, y_S)$. The following security requirements must hold:

Correctness of signature generation: *All signatures on any message generated by any honest group member using GSig will get accepted by the verification algorithm.*

Anonymity and unlinkability and of signatures: *Given two signature-message pairs, it is only feasible to the revocation manager to find out which group member(s) generated any of the signatures or whether the signatures have been generated by the same group member.*

Unforgeability of signatures: *It is feasible to sign messages only to group members (i.e., users that have run the registration protocol with the membership manager) or to the membership manager herself*[2].

Unforgeability of tracing: *The revocation manager cannot accuse a group member falsely of having originated a given signature.*

No framing: *No coalition of group members, the revocation manager, and the membership manager can produce a signature that will be associated to a group member not part of the coalition.*

Unavoidable traceability: *No coalition of group members and the revocation manager (but excluding the membership manager) can generate a valid signature that, when its anonymity is revoked, cannot be associated to a group member.*

To achieve strong or perfect separability it is necessary that the four algorithms GKG-MM, GKG-RM, GKG-GM, and GKG-S can be run completely independent of each other. Moreover, if one of them is re-run the others need not.

The actual algorithm to set-up the group signature scheme is *GKG-S* and should be carried out by representatives of the group members and the membership manager(s) or by a single party they all fully trust. The other three key-generation algorithms are in principle key-generation algorithms of some cryptographic primitives and could have been run in advance of the set-up of the group signature scheme, i.e., they could in principle be excluded from the model.

3 Preliminaries and Proof Techniques

This section summarizes various known results on proofs of knowledge of and about discrete logarithms, combines them into new building blocks, and provides notation for such protocols. In particular, we present new protocols to prove the equality of two discrete logarithms in *different* groups and, more general, the

[2] The membership manager can always invent a fake identity, i.e., run GKG-GM, and issue a certificate on that public key. It is understood that if the revocation lead to a public key not present in *GML* then the membership manager is accused.

validity of polynomial relations in $\mathbb{Z}$ among discrete logarithms from *different* groups.

In the following we assume different groups $G = \langle g \rangle, G_i = \langle g_i \rangle$ $(i = 1, 2)$ of large known prime orders q and q_i, respectively. Furthermore, let h and h_i also be generators of G and G_i, respectively, such that $\log_g h$ and $\log_{g_i} h_i$ are not known. For technical reasons, we define a discrete logarithm $\log_g h$ to be the integer x with $-q/2 < x < q/2$ such that $y = g^x$ and assume that arithmetic modulo q is carried out with this in mind. All protocols exposed in the following are two-party protocols between a male prover and a female verifier.

3.1 Basic Zero-Knowledge Proofs of Knowledge

The most basic protocol is a zero-knowledge proof of knowledge of the discrete logarithm of some group element $y \in G$ to the base g [15,45]. We shortly recall this protocol and its properties: The prover knowing $x = \log_g y$ sends the verifier the *commitment* $t := g^r$, where $r \in_R \mathbb{Z}_q$. Then, the verifier sends the prover a random *challenge* $c \in_R \{0,1\}^k$ to which the prover responds with $s := r - cx \pmod q$. (The integer $k \geq 1$ is a security parameter.) The verifier accepts if $t = g^s y^c$. Triples (t, c, s) with $t = g^s y^c$ are called *accepting* triples. Since this $x = \log_g y$ can be computed from two accepting triples (t, c, s) and $(t, \dot{c}, \dot{s})$ with $c \neq \dot{c}$, i.e., $x := (s - \dot{s})(\dot{c} - c)^{-1} \pmod q$, this protocol is a proof of knowledge of $\log_g y$ when sequentially repeated sufficiently many times. Furthermore, the protocol is honest-verifier zero-knowledge[3] and for $k = \mathcal{O}(\log \log q)$ zero-knowledge for any verifier. Using notation from [11], this protocol is denoted $PK\{(\alpha) : y = g^\alpha\}$, which can be read as "*zero-knowledge* P*roof of* K*nowledge of a value* α *such that* $y = g^\alpha$ *holds.*" The convention is that Greek letters denote the knowledge proven while all other parameters are known to the verifier.

This basic protocol can be extended to prove the knowledge of a representation of a group element $y \in G$ with respect to several bases $z_1, \ldots, z_v$ [15] which is denoted $PK\{(\alpha_1, \ldots, \alpha_v) : y = z_1^{\alpha_1} \cdot \ldots \cdot z_v^{\alpha_v}\}$. Another extension allows to prove the equality of the discrete logarithms of two group elements $y_1, y_2 \in G$ to the bases g and h, respectively [14,16]. The idea is to carry out the basic protocol for y_1 and for y_2 in parallel and requiring that the challenges (c) and the responses (s) are the same [16]. If the verifier accepts, the two logarithms must be equal as can be seen by considering how the prover's secret can be derived from two accepting triples with the same commitment. Such a protocol will be denoted $PK\{(\alpha) : y_1 = g^\alpha \wedge y_2 = h^\alpha\}$. This technique can be generalized to prove equalities among representations of the elements $y_1, \ldots, y_w$ to bases $g_1, \ldots, g_v$ [11]. As an example, the protocol $PK\{(\alpha, \beta) : y_2 = g^\alpha \wedge y_3 = g^\beta \wedge y_1 = y_2^\beta\}$ allows to prove that the discrete logarithm of y_1 is the product of the discrete logarithms of y_2 and y_3 modulo the group's order.

These kinds of proofs of knowledge (PK) can be turned into signature schemes by the so-called Fiat-Shamir heuristic [23]. That is, the prover determines the

[3] Honest verifier zero-knowledge proofs can be made zero-knowledge by requiring the verifier to commit to the challenge before she receives the prover's commitments.

challenge c by applying a collision-resistant hash-function $\mathcal{H}$ to the commitment and the message m that is signed and then computes the response as usual. The resulting signature consists of the challenge and the response. We denote such S*ignature schemes based on a zero-knowledge* Pr*oof of* K*nowledge* (*SPK*) similarly as the *PK*'s, e.g., $SPK\{(\alpha) : y = g^\alpha\}(m)$. Such *SPK*'s can be proven secure in the random oracle model [4,41] given the security of the underlying *PK*'s.

3.2 Interval Proofs for Discrete Logarithms

If we restrict ourselves to binary challenges, i.e., set $k = 1$, the basic protocol $PK\{(\alpha) : y = g^\alpha\}$ can be modified to prove not only the knowledge of $\log_g y$ but also that $\log_g y$ lies within some determined interval [9,13], e.g., $-2^\ell < \log_g y < 2^\ell$. More precisely, this is achieved by requiring the prover's response s to satisfy $-2^{\ell-1} < s < 2^{\ell-1}$. However, the prover can only carry out the protocol successfully, if the tighter bound $-2^{(\ell-2)/\epsilon} < \log_g y < 2^{(\ell-2)/\epsilon}$ holds, where $e > 1$ is a security parameter[4], and when choosing the value r to compute the commitment t such that $-2^{\ell-2} < r < 2^{\ell-2}$. This modified protocol is denoted $PK^b\{(\alpha) : y = g^\alpha \ \wedge \ (-2^\ell < \alpha < 2^\ell)\}$, where the b in PK^b reminds that the protocol uses binary challenges, i.e., is not very efficient. A further modification allows to prove that $a - 2^\ell < \log_g y < a + 2^\ell$, where a is a fixed offset [9][5]. Note that such protocols make sense only if the group's order q is larger than $2^{\ell+1}$.

An alternative but less efficient method for proving bounds on a discrete logarithm is to first commit to every bit of $x = \log_g y$ and then to prove that (1) they constitute the binary representation of x and (2) the committed values are either a 0 or a 1. The latter can be done using techniques from [20,44]. This method is linear in the length of x but allows to prove tighter bounds and, when revealing the most significant bit as 1, even the exact bit-length of x.

3.3 Efficient Interval Proofs for Discrete Logarithms

The choice $k = 1$ is not very attractive from an efficiency point of view and thus the quest for more efficient protocols seems natural. Indeed, one can do better under the strong-RSA assumption [3,25] which is the following variation of the well-known RSA assumption [43].

Assumption 1 (Strong RSA). *There exists a probabilistic algorithm K such that $Pr[z \overset{G}{=} u^e \wedge e > 1 : (G, z) := K(1^{\ell_g}), (u, e) := A(G, z)] < 1/p(\ell_g)$ holds for all probabilistic polynomial-time algorithms A, all polynomials $p(\cdot)$, all sufficiently large ℓ_g, where G is a group of order $\approx 2^{\ell_g}$, $z \in G$, and $e \in \mathbb{Z}$.*

In words, this assumption states that there exists a key-generator K that outputs a group G of unknown order and an element $z \in G/\{\pm 1\}$ such that it is infeasible

[4] In fact, the parameter ϵ controls the tightness of the statistical zero-knowledgeness.
[5] Using different techniques, Damgård obtains a similar result [21].

to find a pair $(u, e) \in G \times \mathbb{Z}$ such that $e > 1$ and $u^e = z$. K could be implemented by choosing G as $\mathbb{Z}_n$, where n is an RSA modulus of size $\approx 2^{\ell_g}$, and picking z randomly from $\mathbb{Z}_n$.

Fujisaki and Okamoto [25] show that under this assumption the protocol $PK\{(\alpha) : y = g^\alpha\}$ works also for groups of *unknown* order, e.g., for $G = \mathbb{Z}_n^*$, where n is an RSA modulus. More precisely, it is pointed out that the discrete logarithm x of y to the base g can be computed from two accepting triples (t, c, s) and $(t, \dot{c}, \dot{s})$ without knowing the group's order: Since $t = g^s y^c = g^{\dot{s}} y^{\dot{c}}$ we must have $x(c - \dot{c}) \equiv \dot{s} - s \pmod{\text{ord}(g)}$. If $c - \dot{c}$ does not divide $\dot{s} - s$ in $\mathbb{Z}$, then one can compute a non-trivial root of g (see [25]). However, due to the strong-RSA assumption the latter is infeasible and hence $c - \dot{c}$ must divide $\dot{s} - s$ in $\mathbb{Z}$ with overwhelming probability and we can compute x without knowing the order of g. This argument generalizes to representations w.r.t. several bases.

The fact that $c - \dot{c}$ divides $\dot{s} - s$ in $\mathbb{Z}$ with overwhelming probability allows us to draw conclusions about the size of the prover's secret from the size of his responses [9,13] similarly as in case of binary challenges. In combination with the ideas that led to the protocol $PK\{(\alpha) : y_1 = g^\alpha \wedge y_2 = h^\alpha\}$ (cf. Section 3.1), this technique can also be used to efficiently prove statements about the size of discrete logarithms from groups with *known* orders: Let ℓ denote a length, $\epsilon > 1$ and k be security parameter, and $G = \langle g \rangle$ be a group whose order $q > 2^{\ell+1}$ is known. Let the prover's secret x be an integer such that $-2^{(\ell-2)/\epsilon-k} < x < 2^{(\ell-2)/\epsilon-k}$ and let $y = g^x$. First, the prover and the verifier engage in a (once and for all) set-up phase. The verifier randomly chooses two sufficiently large safe primes, say p_1 and p_2, and computes $n := p_1 p_2$. (The modulus n must be large enough to avoid factoring but its size needs not to depend on ϵ, ℓ, k, or q.) She chooses two random elements h_1 and h_2 from $\mathbb{Z}_n$, sends the prover n, h_1, and h_2, and proves him that n is indeed the product of two safe primes (e.g., by using the techniques from [10]). The prover checks whether $h_1 \neq \pm 1 \pmod{n}$, $h_2 \neq \pm 1 \pmod{n}$, $\gcd(h_1, n) = 1$, and $\gcd(h_2, n) = 1$ holds. This will convince him that h_1 and h_2 have large order (see [26]). This concludes the set-up phase. Now, the prover chooses $r \in_R \mathbb{Z}_n$, computes $\tilde{y} = h_1^r h_2^x$, and sends the verifier $\tilde{y}$. Finally, they engage in the protocol $PK\{(\alpha, \beta) : y \stackrel{G}{=} g^\alpha \wedge \tilde{y} \stackrel{\mathbb{Z}_n^*}{=} h_1^\beta h_2^\alpha \wedge (-2^\ell < \alpha < 2^\ell)\}$. If they finish the protocol successfully, the verifier will be convinced that the prover knows a value, say x, such that $y = g^x$ and $-2^\ell < x < 2^\ell$ holds.

3.4 Proving Relations among Discrete Logs from Different Groups

The protocols for proving that a discrete logarithms lies within some determined interval, as exposed in the previous two subsections, are a powerful tool. For instance, as is shown in [10], they allow to prove that a discrete logarithm is the product (or sum) of two other discrete logarithms in $\mathbb{Z}$, i.e., *not* modulo the group's order. This extends naturally to the validity of arbitrary polynomial equations over $\mathbb{Z}$ in the prover's secrets. Moreover, these protocols allow to prove the equality of discrete logarithms from *different* groups as is explained in the next paragraph. We believe this is a new building block of independent interest,

which in combination with the results from [10] allows to prove the validity of polynomial relations (in $\mathbb{Z}$) among discrete logarithms from different groups.

Let $G_1 = \langle g_1 \rangle$ and $G_2 = \langle g_2 \rangle$ be two distinct groups of orders q_1 and q_2, respectively, and let ℓ be a integer such that $2^{\ell+1} < \min\{q_1, q_2\}$ holds. Let $y_1 = g_1^x$ and $y_2 = g_2^x$. If x lies in between $-2^{(\ell-2)/\epsilon}$ and $2^{(\ell-2)/\epsilon}$, where $e > 1$ is a security parameter (cf. Section 3.2), the prover can convince the verifier that $\log_{g_1} y_1 = \log_{g_2} y_2$ (in $\mathbb{Z}$) by carrying out $PK^b\{(\alpha) : y_1 \overset{G_1}{=} g_1^\alpha \wedge y_2 \overset{G_2}{=} g_2^\alpha \wedge (-2^\ell < \alpha < 2^\ell)\}$ with her. Since this protocol uses binary challenges it is not very efficient. However, under the strong RSA assumption and if x lies in between $-2^{(\ell-2)/\epsilon-k}$ and $2^{(\ell-2)/\epsilon-k}$, where k denotes the number of bits of the challenge, the prover can efficiently convince the verifier that $\log_{g_1} y_1 = \log_{g_2} y_2$ holds as follows. First the prover and the verifier engage in the (once and for all) set-up phase to generate the modulus n and the elements h_1 and h_2 as described in the previous subsection. Then the prover chooses $r \in_R \mathbb{Z}_n$, computes $\tilde{y} = h_1^r h_2^x \pmod n$, sets it to the verifier, and carries out $PK\{(\alpha, \beta) : y_1 \overset{G_1}{=} g_1^\alpha \wedge y_2 \overset{G_2}{=} g_2^\alpha \wedge \tilde{y} \overset{\mathbb{Z}_n^*}{=} h_1^\beta h_2^\alpha \wedge (-2^\ell < \alpha < 2^\ell)\}$ together with the verifier. We describe this protocol in detail:

1. The prover picks $r_1 \in_R \{-2^{\ell-2}, \ldots, 2^{\ell-2}\}$ and $r_2 \in_R \{-(n2^k)^\epsilon, \ldots, (n2^k)^\epsilon\}$ and computes the commitments $t_1 := g_1^{r_1}$, $t_2 := g_2^{r_1}$, $t_3 := h_1^{r_2} h_2^{r_1}$. He sends the verifier (t_1, t_2, t_3).
2. The verifier returns a random challenge $c \in_R \{0,1\}^k$.
3. The prover computes the responses $s_1 = r_1 - cx$ and $s_2 = r_2 - cr$ (both in $\mathbb{Z}$) and sends the verifier (s_1, s_2).
4. The verifier accepts if and only if $-2^{\ell-1} < s_1 < 2^{\ell-1}$, $t_1 = g_1^{s_1} y_1^c$, $t_2 = g_2^{s_1} y_2^c$, and $t_3 = h_1^{s_2} h_2^{s_1} \tilde{y}^c$ hold.

Clearly, this kind of protocol generalizes to several different groups, to representations, and to arbitrary modular relations.

4 A Generic Separable Group Signature Scheme

This section provides the definitions of some cryptographic primitives and then presents a generic separable group signature scheme based on these primitives.

4.1 Definition of Some Cryptographic Primitives

The first primitive is a *shadow encryption scheme (SE)*. It consists of three algorithms *EKG*, *Enc*, and *Dec* for key generation, encryption, and decryption, respectively. Participants are a sender and a receiver. On input of a security parameter, *EKG* outputs the key pair (x, y) of the receiver. On input of the receiver's public key y and a message m, *Enc* outputs the ciphertext c. On input of the public key y, the secret key x, and the ciphertext c, *Dec* outputs a value v, called the *shadow*. Unlike as for ordinary public key encryption schemes, decryption does not reveal the encrypted message m but only a value v such that the pair (m, v) satisfies a predefined binary relation $\mathcal{R}$.

Definition 2. *Let ℓ be a security parameter and $\mathcal{R} \subseteq \{0,1\}^\ell \times \{0,1\}^*$ a binary one-to-one relation that can be recognized in polynomial time. A triple (EKG, Enc, Dec) of probabilistic polynomial-time algorithms is a secure public-key shadow encryption scheme with respect to $\mathcal{R}$ if the following properties hold.*

Correctness: *For every $(x,y) \in EKG(1^\ell)$ and all $m_1, m_2 \in \{0,1\}^\ell$ we have $(m_1, Dec(y, x, Enc(y, m_2))) \in \mathcal{R}$ if and only if $m_1 = m_2$.*

Security: *For all probabilistic polynomial-time algorithms T and M, all polynomials $p(\cdot)$, and all sufficiently large ℓ we have*

$$\Pr[T(1^\ell, y, m_0, m_1, d) = i : (x,y) := EKG(1^\ell); (m_0, m_1) := M(y, 1^\ell); \\ i \in_R \{0,1\}; d := Enc(y, m_i)] < 1/2 + 1/p(\ell) \ .$$

Furthermore, as encryption and decryption should be efficient, for all messages m values v such that $(m,v) \in \mathcal{R}$ must be efficiently computable. Any semantically secure public key encryption scheme (e.g., [29,37]) will give an shadow encryption scheme with basically the same efficiency. However, if $\mathcal{R}$ is a hard relation (i.e., given v it is infeasible to find an m such that $(m,v) \in \mathcal{R}$), the converse seems to be possible only at a loss of efficiency, e.g., by shadow-encrypting every bit of the message separately. Hence, a shadow encryption scheme is a weaker primitive from an efficiency point of view but is sufficient in applications where full encryption is not necessary. A concept similar to shadow encryption is confirmer commitments [35].

The rest of the primitives are rather standard and therefore we introduce them only informally (for formal definitions see, e.g., [28]). Let f be a one-way function. Let $SIG = (SKG, Sig, Ver)$ denote a signature scheme, where SKG is the key-generation algorithm (that on input 1^ℓ outputs a key pair (x,y)), Sig is the signing algorithm (that on input of a secret key x, the corresponding public key y, and a message m outputs a signature s on m), and Ver is the verification algorithm (that on input of a public key y, an alleged signature s, and a message m outputs 1 if and only if s is a signature on m with respect to y). We require that the signature scheme is existentially unforgeable under a certain kind of chosen-message attack, that is the attacker gets only signatures on messages of which he knows a pre-image under some predetermined one-way function f. Finally, we need an unconditionally hiding commitment scheme, i.e., a function Com that takes as input a string x to commit to and a random string r. One can commit to a value x by $C := Com(x,r)$, where r is randomly chosen. We require that the distribution of C committing to different x's are statistically indistinguishable, i.e., x is information theoretically hidden from the receiver. Furthermore, it should be hard to open a commitment in two ways, i.e., hard to find strings x, $x' \neq x$, r, and r' such that $Com(x,r) = Com(x',r')$

4.2 A Generic Realization of a Separable Group Signature Scheme

This section describes a generic separable group signature scheme and shows its security. The construction extends ideas from [11,12]. Let $SE = (EKG, Enc, Dec)$

be a probabilistic shadow encryption scheme, $SIG = (SKG, Sig, Ver)$ a signature scheme, f a one-way function, and Com an unconditionally hiding commitment scheme. With these primitives a generic separable group signature scheme can be constructed as follows.

GKG-MM: This is the key generation algorithm SKG of the signature scheme SIG.

GKG-RM: This is the key generation algorithm EKG of the probabilistic shadow encryption scheme SE.

GKG-GM: The group member chooses a random value x_U from the domain of f as secret key and computes his public key $y_U = f(x_U)$.

GKG-S: This algorithm chooses a hash function $\mathcal{H}$ suitable for use in the SPK's and sets up a commitment scheme Com.

Reg: The group member sends (ID_U, y_U) to the membership manager and proves in zero-knowledge that he knows x_U such that $y_U = f(x_U)$ holds. If he is successful, the membership manager computes the signature $s_U := Sig(x_M, y_M, y_U)$ and sends it to U. The tuple (y_U, ID_U) is added to GML. The group member's output is s_U.

GSig: To sign a message $m \in \{0,1\}^*$ a group member U computes $z := Enc(y_R, y_U)$, $C := Com(y_U, r)$, where r is a random string, and the three SPK's (using informal notation)

$$\begin{aligned} S_I &:= SPK\{(\alpha,\beta) : C = Com(\alpha,\beta) \ \wedge \ z = Enc(y_R,\alpha)\}(C,z,m) \\ S_{II} &:= SPK\{(\alpha,\beta,\delta) : C = Com(\alpha,\beta) \ \wedge \ \alpha = f(\delta)\}(S_I) \\ S_{III} &:= SPK\{(\alpha,\beta,\gamma) : C = Com(\alpha,\beta) \ \wedge \ Ver(y_M,\gamma,\alpha) = 1\}(S_{II}) \ . \end{aligned}$$

The signature on m is the tuple $(z, C, S_I, S_{II}, S_{III})$.

GVer: A group-signature $\sigma = (z, C, S_I, S_{II}, S_{III})$ on a message m can be verified by checking the SPK's S_I, S_{II}, and S_{III}.

GTrace: Given $(z, C, S_I, S_{II}, S_{III})$ and m, the revocation manager checks whether the signature is valid, shadow-decrypts z as $v = Dec(y_R, x_R, z)$, finds an (y_U, ID_U) from GML such that $(y_U, v) \in \mathcal{R}$, and computes $V := SPK\{(\alpha) : v = Dec(y_R, \alpha, z)\}(m, \sigma, v, y_U)$.

Theorem 1. *In the random oracle model, the above construction is a secure group signature scheme, provided that the requirements of the used commitment scheme, shadow encryption scheme, the one-way function, and the signature scheme are satisfied.*

Proof (sketch). The correctness of the signature generation is obvious.

Anonymity and unlinkability of signatures: Any two different signatures $(z, C, S_I, S_{II}, S_{III})$ and $(z', C', S'_I, S'_{II}, S'_{III})$ are computationally indistinguishable due to the security requirement for the shadow-encryption scheme, the hiding property of the commitment scheme, and the zero-knowledge property of the proofs underlying the SPK's S_I, S_{II}, and S_{III}.

Unforgeability of group signatures: Assuming the soundness of the *SPK*'s S_I, S_{II}, and S_{III} and the commitment scheme's security, a non-group-member being able to forge signatures is also able to compute values $s_{\tilde{U}}$, $y_{\tilde{U}}$, and $x_{\tilde{U}}$ such that $y_{\tilde{U}} = f(x_{\tilde{U}})$ and $Ver(y_M, s_{\tilde{U}}, y_{\tilde{U}}) = 1$. However, the latter implies that the attacker must have existentially forged a signature of the membership manager which is assumed to be infeasible.

Unforgeability of tracing: If the revocation manager could claim that another y_U than the one whose shadow-encryption is contained in a valid signature, either the correctness property of the shadow encryption scheme or the soundness of the *SPK* V would not hold.

No framing: Assume that some coalition can sign on behalf of a group member with membership key y_U. Given the unforgeability of tracing, y_U must be shadow-encrypted in z. Due to the *SPK*'s S_I and S_{II} the coalition must be able to compute $f^{-1}(y_U)$ or to break the commitment scheme, i.e., open a commitment in two ways. Both is assumed to be infeasible.

Unavoidable traceability: Based on the same arguments as in the case of unforgeability of group signatures, we can conclude that a successfully attacking coalition must be able to forge signatures of the membership manager under a known signature-message pair attack, which is assumed to be infeasible. □

This scheme indeed achieves perfect separability since the algorithms GKG-MM, GKG-RM, GKG-GM, and GKG-S can be run independent of each other. For all procedures but GSig it follows from the construction that they are efficient if the underlying primitives are efficient. To achieve efficiency for GSig as well, it seems necessary to restrict the choices of the instances of the cryptographic primitives in which case we get group signature scheme with strong separability. We refer to the next section for possible instances of the employed primitives.

It can easily be seen that the size of the group's public key and the length of signatures do not depend on the number of group members. However, in the tracing algorithm *GTrace*, the revocation manager has to check the membership key of every group member, hence the running-time of this algorithm is linear in the number of group members. We will later see that in our implementation this can be overcome and the tracing algorithm can also be made independent from the group's size. This problem could as well be solved by using semantically secure encryption (e.g., [29,37]) instead of shadow encryption. However, finding an instance of the resulting generic group signature scheme with efficient signing and verification procedures is an open problem.

We note that a (generic) identity escrow scheme can be obtained from the above scheme by replacing the *SPK*'s in the signature generation algorithm by the underlying *PK*'s.

5 Instances

This section provides concrete instances of cryptographic primitives that allow an efficient realization of the *SPK*'s in the procedures *GSig* and *GTrace* of our

generic group signature scheme. All these instances are based on the discrete logarithm problem and some are additionally based on the hardness of factoring or computing roots modulo a composite. The somewhat less efficient instances which are solely based on the discrete logarithm problem are presented in the full paper.

Throughout this section, $m \in \{0,1\}^*$ denotes the message that a group member U wants to sign, k and $\epsilon > 1$ are security parameters (k denotes the bit-length of the challenges in the *SPK*'s and ϵ controls the tightness of the zero-knowledgeness, cf. Section 3), 2^{ℓ_U} is an upper-bound on number of elements in the domain and the image of the one-way function $f : \{0,1\}^{\ell_U/2} \times \{0,1\}^{\ell_U/2} \to \{0,1\}^{\ell_U}$, and ℓ_M and ℓ_R are length-parameters of the crypto-systems chosen by the membership and the revocation manager, respectively.

5.1 A Commitment Scheme

As commitment scheme *Com* we apply the one due to Pedersen [39] which is information theoretically hiding and computational binding (i.e., the binding property relies on the hardness of computing discrete logarithms). Deviating from the original proposal, we use this scheme with an algebraic group of large unknown order. This does not alter the scheme's properties, but will allow us to use this group also for the efficient interval-proofs described in Section 3. An example of such a group is a subgroup of $\mathbb{Z}_n^*$, where n is a large RSA modulus whose factors are unknown. Either this modulus is chosen by a trusted third party or by representatives of the group members and the membership manager(s). In the latter case, the parties can employ the protocols presented in [6,24,42] to choose such a modulus jointly without the participants learning its factors. In the following we stick to the latter.

These choices have the following consequences for the affected procedures of our group signature scheme.

GKG-S (commitment part): The representatives of the group members and the membership manager jointly choose an RSA modulus $n_S > 2^{\ell_S}$, such that the factors of n_S are unknown, and a random element $h_S \in \mathbb{Z}_{n_S}^*$ (e.g., using techniques from [6,24,42]). Furthermore, they all choose a random exponent $r_i \in \{0,1\}^{\ell_S}$ and commit to $h_i = h_S^{r_i} \pmod{n_S}$ using some secure commitment scheme. If all commitments are published, they open the commitments, prove their knowledge of $\log_{h_S} h_i$, and compute $g_S = \prod_i h_i$. The parameters n_S, $G_S = \langle h_S \rangle$, g_S, h_S, and ℓ_S are published as part of the group's public key.

GSig (commitment part): A group member can commit to y_U by computing $C := g_S^{y_U} h_S^r$, where r is randomly chosen from $\{-2^{\ell_S}, \ldots, 2^{\ell_S}\}$.

5.2 A Shadow Encryption Scheme

This section provides a shadow encryption scheme that is based upon the ElGamal [22] encryption scheme which we briefly summarize. The public key of the recipient consists of a group $G_R = \langle g_R \rangle$, its prime order q_R , and $y_R = h_R^{x_R}$,

where the recipient's secret key x_R is randomly chosen from $\mathbb{Z}_{q_R}$. The encryption of a message $w \in G$ is a pair $(A := g_R^r, B := wy_R^r)$, where r is randomly chosen from $\mathbb{Z}_{q_R}$. Decryption works by computing $B/A^{x_R} (= w)$. From this scheme a shadow encryption scheme for the relation $\mathcal{R} = \{(u, v) | v = g_R^u\}$ can be derived by encrypting g_R^w instead of w, where now $w \in \mathbb{Z}_{q_R}$. Deciding whether some message $\tilde{w} \in \mathbb{Z}_{q_R}$ is indeed shadow-encrypted in a pair (A, B) can be done (when knowing the secret key) by checking whether $g_R^{\tilde{w}}$ equals B/A^{x_R}. The security properties (cf. Def. 2) of this shadow encryption scheme are inherited from the ElGamal scheme, which is equivalent to the Diffie-Hellman decision problem [46].

With this shadow encryption scheme the affected procedures of our group signature scheme are as follows. Recall that the group member has committed to y_U by $C = g_S^{y_U} h_S^r$ (cf. Section 5.1).

GKG-RM: The revocation manager chooses a group $G_R = \langle g_R \rangle$ of order q_R, a random secret key $x_R \in_R \mathbb{Z}_{q_R}$, computes $y_R := g_R^{x_R}$, and publishes (y_R, g_R, G_R, q_R) as her public key. Let $\ell_R = \lfloor \log_2 q_R \rfloor$.

Part I of GSig: In the following, we assume that[6] $\ell_R > (\ell_U + k)\epsilon + 2$ holds. A group member U shadow-encrypts y_U by computing $A := g_R^r$ and $B := g_R^{y_U} y_R^r$. He then can compute the first *SPK* as

$$S_I := SPK\{(\alpha, \beta, \gamma) : A \overset{G_R}{=} g_R^\alpha \ \wedge \ B \overset{G_R}{=} g_R^\beta y_R^\alpha \ \wedge \ C \overset{G_S}{=} g_S^\beta h_S^\gamma \ \wedge \ (-2^{\ell_R} < \beta < 2^{\ell_R})\}(A, B, C, m) \ .$$

The *SPK* S_I shows that the value committed to by C is indeed shadow-encrypted in (A, B) under y_R.

GTrace: Knowing x_R the revocation manager can check whether some $y_{\tilde{U}}$ is shadow-encrypted in a pair (A, B) that is part of a valid group signature σ on m by testing if $g_R^{y_{U'}} \overset{G_R}{=} B/A^{x_R}$ holds. If it does, she can prove this by

$$V := SPK\{(\alpha) : y_R \overset{G_R}{=} g_R^\alpha \ \wedge \ B/g_R^{y_{\tilde{U}}} \overset{G_R}{=} A^\alpha \ \}(m, \sigma, g_R^{y_{\tilde{U}}}, y_{\tilde{U}}) \ .$$

Remark 1. The tracing algorithm can be made independent of the number of group members if the "shadows" $g_R^{y_U}$ are stored along with y_U, ID_U, and s_U. Then, tracing can be done with a single look-up in the database. This has of course the disadvantage, that the database must be updated if the revocation manager changes her public key.

5.3 A One-Way Function $f(\cdot)$

Let the function $f : \texttt{primes}^{\ell_U/2} \times \texttt{primes}^{\ell_U/2} \to \{0,1\}^{\ell_U}$ be the multiplication of two $\ell_U/2$-bit primes, i.e., $f(p'_U, p''_U) := p'_U p''_U$. For large enough ℓ_U, this function is believed to be one-way.

[6] For a solution for the case $\ell_R < (\ell_U + k)\epsilon + 2$ we refer to the full paper.

GKG-GM: Group member U chooses two suitable primes p'_U and p''_U with $2^{\ell_U/2-1} < p'_U, p''_U < 2^{\ell_U/2}$ and computes $y_U = p'_U p''_U$. He publishes his public key y_U and keeps (p'_U, p''_U) as his secret key.

Part II of GSig: Given the factors p'_U and p''_U of y_U, where y_U is committed to by C (cf. Section 5.1), group member U can compute the third *SPK* as follows. He picks $r_1, r_2 \in_R \{-2^{\ell_S}, \ldots, 2^{\ell_S}\}$, computes $F := g_S^{p'_U} h_S^{r_1}$, $L := g_S^{p''_U} h_S^{r_2}$, and

$$S_{II} := SPK\big\{(\omega, \theta, \nu, \psi, \kappa) : F \overset{G_S}{=} g_S^{\omega} h_S^{\theta} \ \wedge\ L \overset{G_S}{=} g_S^{\nu} h_S^{\psi} \ \wedge\ C \overset{G_S}{=} F^{\nu} h_S^{\kappa} \ \wedge\ (-2^{(\ell_U/2+k)\epsilon+2} < \omega, \nu < 2^{(\ell_U/2+k)\epsilon+2})\big\}(S_I, F, L)\ .$$

This *SPK* shows that the integer committed to by C is the product (in $\mathbb{Z}$) of the two integers committed to by F and L. Assuring that F and L are nontrivial factors of the integer committed to by C requires $(\ell_U/2+k)\epsilon+2 < \ell_U$ and that the membership manager signs only y_U's that lie between 2^{ℓ_U-1} and 2^{ℓ_U}.

5.4 A Signature Scheme

Signature schemes that are applicable must allow an efficient proof of knowledge of a membership manager's signature on y_U's that are committed to by some $C = Com(y_U)$ (cf. Section 5.1). Typically, signature schemes require the use of a hash function as redundancy function to be existentially unforgeable. However, the need for efficient proofs requires that the redundancy function allows to compute the commitment $C' = Com(red(y_U))$ given C only. An example that allows this is $red(x) = x2^K + d$, where a randomly chosen $d \in \{0, \ldots, 2^K - 1\}$ is fixed and K is a security parameter. Note that some attacks on RSA with such simple redundancy schemes are known [27,36]. However, these attacks seem not to work, if K is chosen sufficiently large. Moreover, they are (arbitrary) chosen-message attacks and are therefore not applicable as our construction requires only a signature scheme that is existential unforgeable under a restricted kind of chosen message attack (cf. Section 4.1). Hence, the RSA signature scheme together with this simple redundancy function seems to satisfy our requirements.

In the following we assume ℓ_U and ℓ_M are such that $\ell_U + K \leq \ell_M$ holds. Using the RSA signature scheme [43] together with this redundancy function has the following consequences.

GKG-MM: The membership manager chooses two $\ell_M/2$-bit primes p'_M and p''_M, computes $n_M = p'_M p''_M$, chooses a prime $e_M > 1$, selects a random integer $d \in_R \{0, \ldots, 2^K - 1\}$, publishes (n_M, e_M, d) as her public key, and stores (p'_M, p''_M) as her secret key.

Reg: The group member sends (y_U, ID_U) to the membership manager and proves her (1) that y_U is the product of two primes and (2) that these primes are of size $\approx 2^{\ell_U/2}$. The first can be done with protocols from [7,10,47]; for the latter the group member computes $c_{2'} := g_S^{p'_U} h_S^{v_{p'}}$ and $c_{2''} := g_S^{p''_U} h_S^{v_{p''}}$, where

$v_{p''}, v_{p'} \in_R \{-2^{\ell_S}, \ldots, 2^{\ell_S}\}$, and carries out the protocol

$$PK^b\{(\alpha,\beta,\gamma,\delta,\kappa) : c_{2'} \overset{G_S}{=} g_S^{\alpha} h_S^{\beta} \ \wedge \ c_{2''} \overset{G_S}{=} g_S^{\gamma} h_S^{\delta} \ \wedge \ g_S^{y_U} \overset{G_S}{=} c_{2'}^{\gamma} h_S^{\kappa} \ \wedge \ (-2^{\epsilon\ell_U/2+2} \leq \alpha, \gamma \leq 2^{\epsilon\ell_U/2+2})\}$$

with the membership manager. If all these proofs are fine and if y_U is an ℓ_U-bit number, the membership manager signs y_U, i.e., computes $s_U := red(y_U)^{1/e_M} \equiv (y_U 2^K + d)^{1/e_M} \pmod{n_M}$, and sends s_U to the group member who checks its validity. Finally, the two parties enter (y_U, ID_U) in the membership list *GML*.

Part III of GSig: We assume that $e_M = 3$ (other cases can be done similarly). Group member U computes $D := g_S^{s_U} h_S^{r_1}$, $E := g_S^{s_U^2 \pmod{n_M}} h_S^{r_2}$, with $r_1, r_2 \in_R \{-2^{\ell_S}, \ldots, 2^{\ell_S}\}$, and

$$S_{III} := SPK\{(\beta,\gamma,\psi,\lambda,\tau,\pi,\delta,\zeta,\rho,\xi) : C \overset{G_S}{=} g_S^{\beta} h_S^{\gamma} \ \wedge \ D \overset{G_S}{=} g_S^{\psi} h_S^{\lambda} \ \wedge \ E \overset{G_S}{=} g_S^{\tau} h_S^{\pi} \ \wedge \ E \overset{G_S}{=} D^{\psi} (g_S^{n_M})^{\delta} h_S^{\zeta} \ \wedge \ C^{2^K} g_S^{d} \overset{G_S}{=} E^{\psi} (g_S^{n_M})^{\rho} h_S^{\xi} \ \wedge \ (-2^{(\ell_U+k)\epsilon+2} < \beta < 2^{(\ell_U+k)\epsilon+2}) \ \wedge \ (-2^{(\ell_M+k)\epsilon+2} < \psi, \tau, \delta, \rho < 2^{(\ell_M+k)\epsilon+2})\}(S_I, S_{II}, D, E) \ .$$

This *SPK* shows that the cubicle of the integer committed to by D equals the integer obtained when applying the redundancy function *red* to integer committed to by C (modulo n_M).

5.5 Efficiency Considerations and Remarks

To make the efficiency consideration easier, we first put the different parts of the signature generation algorithm together. Furthermore, we merge the *SPK*'s S_I, S_{II}, and S_{III} into a single one.

GSig (all parts): Knowing $y_U = p'_U p''_U$, $x_U = (p'_U, p''_U)$, and s_U, group member U can sign a message $m \in \{0,1\}^*$ on the group's behalf by choosing $r_1, r_3, r_4, r_5, r_6 \in_R \{-2^{\ell_S}, \ldots, 2^{\ell_S}\}$ and $r_2 \in_R \mathbb{Z}_{q_R}$ and computing $A := g_R^{r_2}$, $B := g_R^{y_U} y_R^{r_2}$, $C := g_S^{y_U} h_S^{r_1}$, $D := g_S^{s_U} h_S^{r_3}$, $E := g_S^{s_U^2 \pmod{n_M}} h_S^{r_4}$, $F := g_S^{p'_U} h_S^{r_5}$, $L := g_S^{p''_U} h_S^{r_6}$, and

$$S_{I-III} := SPK\{(\alpha,\beta,\gamma,\psi,\lambda,\tau,\pi,\delta,\zeta,\omega,\theta,\nu,\mu,\rho,\xi,\kappa) : A \overset{G_R}{=} g_R^{\alpha} \ \wedge \ B \overset{G_R}{=} g_R^{\beta} y_R^{\alpha} \ \wedge \ C \overset{G_S}{=} g_S^{\beta} h_S^{\gamma} \ \wedge \ D \overset{G_S}{=} g_S^{\psi} h_S^{\lambda} \ \wedge \ E \overset{G_S}{=} g_S^{\tau} h_S^{\pi} \ \wedge \ E \overset{G_S}{=} D^{\psi} (g_S^{n_M})^{\delta} h_S^{\zeta} \ \wedge \ F \overset{G_S}{=} g_S^{\omega} h_S^{\theta} \ \wedge \ L \overset{G_S}{=} g_S^{\nu} h_S^{\mu} \ \wedge \ C^{2^K} g_S^{d} \overset{G_S}{=} E^{\psi} (g_S^{n_M})^{\rho} h_S^{\xi} \ \wedge \ C \overset{G_S}{=} F^{\nu} h_S^{\kappa} \ \wedge \ (-2^{\epsilon(\ell_U+k)+2} < \beta < 2^{\epsilon(\ell_U+k)+2}) \wedge (-2^{\epsilon(\ell_M+k)+2} < \psi, \tau, \delta, \rho < 2^{\epsilon(\ell_M+k)+2}) \wedge (-2^{\epsilon(\ell_U/2+k)+2} < \omega, \nu < 2^{\epsilon(\ell_U/2+k)+2})\}(A, B, C, D, E, F, L, m) \ .$$

Provided that elements from G_S and G_R have roughly the same size, the signer's computational load is about 17 multi-exponentiations and the verifier's computational effort is about 10 multi-exponentiations. If elements in G_R and G_S are about ℓ_R and ℓ_S bits long, respectively, a signature takes $5\ell_S + 2\ell_R + \ell_R + 2(\epsilon(\ell_U/2+k)+2) + (\epsilon(\ell_U+k)+2) + 8(\epsilon(\ell_S+k)+2) + 4(\epsilon(\ell_M+k)+2) + k$ bits. The following choices of security parameter (such that $\ell_R > \epsilon(\ell_U + k) + 2$ (cf. Section 5.2), $(\ell_U/2 + k)\epsilon + 2 < \ell_U$ (cf. Section 5.3), and $\ell_M > \ell_U + K$ (cf. Section 5.4)) could be used: $\epsilon = 9/8$, $\ell_U = 768$, $\ell_R = 1100$, $\ell_M = 1630$, $\ell_S = 1024$, $K = 850$, and $k = 160$. With these choices a signature is about $3,5$ kilobytes long. Compared to the most efficient non-separable group signature schemes [9,11,12], we lose roughly a factor of 3 in terms of the length of signatures as well as the number of exponentiations.

6 Extensions and Open Problems

In order to keep the system manageable it is desirable to be able to remove group members from the group. This implies that the group's public key must be changed each time that a group member is removed and also that each group signature is time-stamped. In principle, group members can be removed by changing the membership manager's key and by issuing new certificates to the remaining group members. A more elegant way to realize this is using so-called *one-way accumulators* (OWA) [3,5,19], in particular by using the efficient realization of an OWA given in [3]. Details are provided in the full paper.

Another possible extension is to distribute the role the membership and the revocation manager among several parties while still ensuring the separability with respect to all parties. This can be done by combining standard secret sharing techniques with the proof techniques described in Section 3, in particular those in Section 3.4.

Further research is required for an exact security analysis of the signature scheme presented in Section 5.4. Finding a signature schemes with simple redundancy functions and designing a separable group signature scheme with efficient signature generation and verification that is exclusively based on standard assumptions are challenging open problems.

Acknowledgments

The authors are grateful to Ivan Damgård and Louis Salvail for discussions.

References

1. G. Ateniese and G. Tsudik. Group signatures à la carte. In *ACM Symposium on Discrete Algorithms*, 1999.
2. G. Ateniese and G. Tsudik. Some open issues and new directions in group signatures. In *Proc. of Financial Cryptography '99*, 1999.

3. N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology — EUROCRYPT '97*, vol. 1233 of *LNCS*, pp. 480–494.
4. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM CCS*, pp. 62–73. ACM, 1993.
5. J. Benaloh and M. de Mare. One-Way Accumulators: A Decentralized Alternative to Digital Sinatures. In *Advances in Cryptology — EUROCRYPT '93*, vol. 765 of *LNCS*, pp. 274-285.
6. D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In *Advances in Cryptology — CRYPTO '97*, vol. 1296 of *LNCS*, pp. 425–439.
7. J. Boyar, K. Friedl, and C. Lund. Practical zero-knowledge proofs: Giving hints and using deficiencies. *Journal of Cryptology*, 4(3):185–206, 1991.
8. J. Camenisch. Efficient and generalized group signatures. In *Advances in Cryptology — EUROCRYPT '97*, vol. 1233 of *LNCS*, pp. 465–479.
9. J. Camenisch and M. Michels. A group signature scheme based on an RSA-variant. Tech. Rep. RS-98-27, BRICS, Dept. of Comp. Sci., University of Aarhus, preliminary version in *Advances in Cryptology — ASIACRYPT '98*, vol. 1514 of *LNCS*.
10. J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *Advances in Cryptology — EUROCRYPT '99*, vol. 1592 of *LNCS*, pp. 107–122.
11. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology — CRYPTO '97*, vol. 1296 of *LNCS*, pp. 410–424.
12. J. L. Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem*. PhD thesis, ETH Zürich, 1998.
13. A. Chan, Y. Frankel, and Y. Tsiounis. Easy come – easy go divisible cash. GTE Technical Report, preliminary version appeared in *Advances in Cryptology — EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 561–575.
14. D. Chaum. Zero-knowledge undeniable signatures. In *Advances in Cryptology — EUROCRYPT '90,* vol. 473 of *LNCS*, pp. 458–464.
15. D. Chaum, J.-H. Evertse, and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In *Advances in Cryptology — EUROCRYPT '87*, vol. 304 of *LNCS*, pp. 127–141.
16. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Advances in Cryptology — CRYPTO '92*, vol. 740 of *LNCS*, pp. 89–105.
17. D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology — EUROCRYPT '91*, vol. 547 of *LNCS*, pp. 257–265.
18. L. Chen and T. P. Pedersen. New group signature schemes. In *Advances in Cryptology — EUROCRYPT '94*, vol. 950 of *LNCS*, pp. 171–181.
19. R. Cramer. Personal communication.
20. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology — CRYPTO '94*, volume 839 of *LNCS*, pp. 174–187. Springer Verlag, 1994.
21. I. B. Damgård. Practical and provable secure release of a secret and exchange of signatures. In *Advances in Cryptology — EUROCRYPT '93*, vol. 765 of *LNCS*, pp. 200–217.
22. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology — CRYPTO '84*, vol. 196 of *LNCS*, pp. 10–18.
23. A. Fiat and A. Shamir. How to prove yourself: Practical solution to identification and signature problems. In *Advances in Cryptology — CRYPTO '86*, vol. 263 of *LNCS*, pp. 186–194.

24. Y. Frankel, P. D. MacKenzie, and M. Yung. Robust efficient distributed RSA-key generation. In STOC'98, pp. 663-672, 1998.
25. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology — CRYPTO '97*, vol. 1294 of *LNCS*, pp. 16–30.
26. R. Gennaro, H. Krawczyk, and T. Rabin. RSA-based undeniable signatures. In *Advances in Cryptology — CRYPTO '97*, vol. 1296 of *LNCS*, pp. 132–149.
27. M. Girault and J.-F. Misarsky. Selective forgery of RSA using redundancy In *Advances in Cryptology — EUROCRYPT '97*, vol. 1233 of *LNCS*, pp. 495-507.
28. S. Goldwasser and M. Bellare. Lecture notes on cryptography, June 1997.
29. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, Apr. 1984.
30. J. Kilian and E. Petrank. Identity escrow. In *Advances in Cryptology — CRYPTO '98*, vol. 1642 of *LNCS*, pp. 169–185.
31. S. J. Kim, S. J. Park, and D. H. Won. Convertible group signatures. In *Advances in Cryptology — ASIACRYPT '96*, vol. 1163 of *LNCS*, pp. 311–321.
32. W.-B. Lee and C.-C. Chang. Efficient group signature scheme based on the discrete logarithm. *IEE Proc. Comput. Digit. Tech.*, 145(1):15–18, 1998.
33. C. H. Lim and P. J. Lee. On the security of convertible group signatures. *Electronics Letters*, 1996.
34. M. Michels. Comments on some group signature schemes. Technical Report TR-96-3-D, Dept. of Comp. Sci., Univ. of Technology, Chemnitz-Zwickau, Nov. 1996.
35. M. Michels and M. Stadler. Generic constructions for secure and efficient confirmer signature schemes. In *Advances in Cryptology — EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 406–421.
36. J.-F. Misarsky. A multiplicative attack using LLL Algorithm on RSA signatures with redundancy. In *Advances in Cryptology — CRYPTO '97*, vol. 1294 of *LNCS*, pp. 221-234.
37. S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM Journal on Computing*, 17(2):412–426, April 1988.
38. S. J. Park, I. S. Lee, and D. H. Won. A practical group signature. In *Proc. of the 1995 Japan-Korea Workshop on Information Security and Cryptography*, 1995.
39. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO '91*, vol. 576 of *LNCS*, pp. 129–140.
40. H. Petersen. How to convert any digital signature scheme into a group signature scheme. In *Security Protocols Workshop*, vol. 1361 of *LNCS*, pp. 177 - 190, 1997.
41. D. Pointcheval. *Les Preuves de Connaissance et leurs Preuves de Sécurité*. PhD thesis, Université de Caen, 1996.
42. G. Poupard and J. Stern. Generation of shared RSA keys by two parties. In *Advances in Cryptology — ASIACRYPT '98*, vol. 1514 of *LNCS*, pp. 11-24.
43. R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. of the ACM*, 21(2):120–126, 1978.
44. A. de Santis, G. di Crescenzo, G. Persiano, and M. Yung. On Monotone Formula Closure of SZK. *35th FOCS*, IEEE, pp. 454–465, 1994.
45. C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.
46. Y. Tsiounis and M. Yung. On the security of ElGamal-based encryption. In *International Workshop on Practice and Theory in Public Key Cryptography*, 1998.
47. J. van de Graaf and R. Peralta. A simple and secure way to show the validity of your public key. In *Advances in Cryptology — CRYPTO '87*, vol. 293 of *LNCS*, pp. 128–134.