

List Signature Schemes

Sébastien Canard^a Berry Schoenmakers^b Martijn Stam^c
Jacques Traoré^a

^a*France Telecom R&D*
42, rue des Coutures, BP6243
14066 Caen Cedex 4, France

^b*Technische Universiteit Eindhoven*
P.O.Box 513, 5600 MB Eindhoven, The Netherlands

^c*Dept. Computer Science, University of Bristol, Merchant Venturers Building,
Woodland Road, Bristol, BS8 1UB, United Kingdom.*

Abstract

A group signature scheme allows members of a group to issue signatures on behalf of the group, while hiding for each signature which group member actually issued it. A group signature scheme also involves a group manager, who is able to open any group signature by showing which group member issued it.

We introduce the concept of list signatures as a variant of group signatures which set a limit on the number of signatures each group member may issue. These limits must be enforced *without* having the group manager open signatures of honest group members—which excludes the trivial solution in which the group manager opens every signature to see whether some group members exceed their limits. Furthermore, we consider the problem of *publicly identifying* group members who exceed their limits, also *without* involving the group manager.

Key words: Electronic voting schemes, Group signature schemes, List signature schemes.

1 Introduction

The basic functionality of a group signature scheme, as introduced by Chaum and Van Heijst (18), is to allow members of a group to issue signatures on behalf of the group, while hiding for each signature which group member

* Discrete Applied Mathematics volume 154-2, pp. 189-201, 2006

actually issued it. In addition it must be possible for a group manager to open any signature issued on behalf of the group, by showing which member issued it.

The group manager is therefore in a powerful position and must be trusted not to revoke the anonymity of group members without permission. To lower the trust in the group manager it may be implemented in a distributed fashion (using threshold cryptography) such that signatures are only opened if a majority of the proxies agrees to do so.

In this paper we study methods to extend the functionality of group signatures while limiting the involvement of the group manager as much as possible. We introduce the notion of *public detection*, which in its simplest form enables any party to detect if a group member attempts to issue more than one signature on behalf of the group. Obviously, this problem can be solved by having the group manager open every signature to check whether a group member signed twice. In that case, however, all signatures by honest group members are *also* opened. Loosely speaking, we define a *list signature scheme* as a group signature scheme with public detection but without opening manager. To prevent the need to continuously rekey the scheme, we let signatures depend on a time frame and only require detection of signatures of the same user within the same time frame (and unlinkability otherwise).

We also introduce the notion of list signatures with *public identification*. This type of scheme not only allows any party to detect dishonest group members, but also for any party to identify them based on their signatures issued on behalf of the group. Hence, the absence of a group manager who can identify misbehaving participants does not actually help these culprits in trying not to be identified. This is particularly useful in an on-line/off-line scenario, where signatures are verified on-line but only checked for double use later.

Group signatures are sometimes associated with applications such as electronic cash and electronic voting. In these applications, a central problem is to prevent payers from spending the same coin twice and voters from casting more than one ballot, respectively. Group signature schemes with public detection substantiate these claims by building in a mechanism to detect multiple signatures by the same group member. As explained above, we require that the scheme can be run in an *optimistic* mode: signatures of honest group members need not be opened; however, if multiple signatures by the same group member are detected, the group manager may reveal the identity by opening one of these signatures.

There are several constructs known in the literature that are related to group signatures, such as identity-escrow, anonymous credentials, concurrent signatures, ring signatures, traceable signatures, and direct anonymous attestation.

We briefly discuss the last three, since they are most relevant to our new proposal of list signatures.

Ring signatures were introduced by Rivest et al. (28) as a light-weight alternative to group signatures. The important feature of a ring signature is the fact that groups are made ad-hoc without the intervention of a group manager by distilling a group public key from the public keys of the intended group members. Ring signatures as originally proposed do not possess a manager to open signatures. Recent work by Dodis et al. (21) show that ring signatures can in fact be equipped with an opening mechanism, in which case they become group signatures with vastly simplified group management. Obviously for list signatures a distinction can also be made between different types of group management.

Traceable signatures were introduced by Kiayias et al. (24). They offer the same functionality as group signatures, but with two added possibilities called tracing and claiming. It is relatively straightforward for a signer to claim ownership of a signature as long as he knows the randomness used to create his signature (and note that, contrary to ordinary signatures, group signatures necessarily need to be randomized since they include an encryption of the identity). Kiayias et al. consider the situation where a signer can claim a signature originated from him without needing to know the randomness. Tracing a signature allows the group manager to publish a value that allows the tracing of all signatures originating from a specific signer.

Direct anonymous attestation was recently introduced by Brickell et al. (11). It is slightly different from the schemes above, in that the user is actually split in two parts: a trusted platform module (TPM) and a host (for instance a mobile phone). If we ignore this separation, direct anonymous attestation can be seen as a group signature without the feature that a signature can be opened, but with the added functionality that signatures originating from the same user can be made linkable.

Roadmap Our first result is presented in Section 3, where we show how to efficiently prove equality of discrete logarithms in the 1-out-of- n setting. In Section 4 we give a definition of what constitutes a list signature scheme and, after introducing the basic design ideas in Section 5, we present a list signature scheme suitable for small groups (of users) in Section 6 and one for large groups in Section 7. We conclude with some remarks about the applicability of list signatures to electronic voting schemes in Section 8. But we begin with introducing some notation and a brief and informal discussion of the relevant hardness assumptions.

2 Intractability Assumptions

The Strong RSA Assumption Let p' and q' be primes of equal length, such that both $p = 2p' + 1$ and $q = 2q' + 1$ are also prime. The number $n = pq$ is known as a safe RSA modulus. The group (under multiplication) of quadratic residues modulo n is denoted by $QR(n)$. It is not too hard to see that $QR(n)$ is a cyclic group of order $p'q'$.

The flexible RSA problem is defined as finding $u \in QR(n)$ and $e \in \mathbb{Z}_{>1}$ such that $u^e = y \pmod n$ when given y and n . Solving the flexible RSA problem is easy for those knowing the factorisation of n , however the strong RSA assumption states that given only y and n , the flexible RSA problem is hard to solve.

The Decision Diffie Hellman Problem Let G be a finite cyclic abelian group with generators f and g . Now suppose someone prepares two samples: in one f and g are raised to the same random power, in the other to two independently random powers (which in effect results in two random elements of G). The Decision Diffie Hellman problem is to distinguish between these two samples, given the four group elements (that is, f, g and the two powers).

The DDH problem has a nice and well-known reducibility property. If it is hard to distinguish tuples (f, g, f^x, g^x) from random tuples (f, g, f^x, g^y) , then it is also hard to distinguish for example tuples (f, g, h, f^x, g^x, h^x) from random tuples (f, g, h, f^x, g^y, h^z) .

If $G = QR(n)$, then G has composite order and for those knowing the group decomposition of G (i.e., knowing the factorisation of n), the DDH problem in G reduces to the DDH problem in the respective components of G . That is, the DDH in G is hard if and only if it is hard in all of its components, where the components of course are regarded as computational objects and not purely group-theoretic ones.

3 Proving Subset Relations for Sets of Discrete Logs

Suppose f, g are generators of a group G_q of prime order q for which $\log_g f$ is unknown. We present an efficient zero-knowledge proof of membership for the language consisting of tuples $(y_1, \dots, y_n, a_1, \dots, a_m) \in G_q^{n+m}$, $1 \leq m \leq n$, satisfying

$$\{\log_f a_j \mid 1 \leq j \leq m\} \subseteq \{\log_g y_i \mid 1 \leq i \leq n\} \quad (1)$$

But before we do so, a quick reminder of the tools we use. Knowledge of a

discrete logarithm can be proven using Schnorr's protocol (29). It takes the prover a single exponentiation to perform, whereas simulating it would require a double exponentiation. Chaum and Pedersen (17) proposed an extension to proof equality (and knowledge) of discrete logarithms. A prover now needs to perform two single exponentiations and the cost of simulating are also doubled. Finally, Cramer, Damgård and Schoenmakers (19) have given a technique to proof one out of a many statements, but without revealing which statement (in fact, their technique is even more general). For the statement that is known, the costs are the same as for the prover of that statement, whereas for the other cases the costs equal that of simulation (hence, if the prover happens to know more than what is required of him to proof, he can speed things up a bit).

We now return to a proof for the language we described. If $m = n = 1$ the language coincides with the language for the Chaum-Pedersen proof described above. For the general case, it is possible to directly apply the technique for proving 1-out-of- n relations using the Chaum-Pedersen proof as the basic proof to show that $\log_f a_j \in \{\log_g y_i \mid 1 \leq i \leq n\}$, for $j = 1, \dots, m$. Indeed this is the approach followed by (13; 20) to construct proofs for similar statements. However, the total work for the proof becomes approximately $2mn$ double exponentiations.

We obtain an improved protocol by breaking up the proof in a different way. As a result we are able to reduce the total work by a factor of two, reducing it to about mn double exponentiations, which seems minimal.

The protocol is based on the following lemma.

Lemma 1. *Suppose $\log_f g$ is unknown. If one proves knowledge of witnesses u_i, v_j, w_j satisfying*

$$y_i = g^{u_i}, \text{ for } i = 1, \dots, n \quad (2)$$

$$a_j = f^{v_j}, \text{ for } j = 1, \dots, m \quad (3)$$

$$\exists_{i=1}^n a_j y_i = (fg)^{w_j}, \text{ for } j = 1, \dots, m \quad (4)$$

then it follows that (1) is satisfied.

Proof. Consider an arbitrary j , and let i be such that $y_i a_j = (fg)^{w_j}$, hence $g^{u_i} f^{v_j} = (fg)^{w_j}$. The particular witnesses can be obtained from the knowledge extractor, but this implies that we can compute $\log_g f$ as

$$\log_g f = (u_i - w_j)/(w_j - v_j),$$

unless $v_j = w_j$, and hence $u_i = v_j = w_j$. Therefore, (1) must hold. \square

Therefore, in order to prove that (1) holds, it suffices to prove that (2)–(4) holds. For (2) and (3) we need m and n Schnorr proofs, respectively. Statement (4) can be proven using the technique for proving 1-out-of- n relations, requiring the work of mn Schnorr simulations.

We will apply Lemma 1 in Section 6 to obtain an efficient list signature scheme. It can also be used to speed up Camenisch’ group signature scheme (13) or multiway elections (20) by a factor of two.

4 List Signatures: Definition

We now move to a new type of signature scheme, which has as defining feature that they are linkable all along a time frame. More precisely, if a single user signs twice within the same time frame, his two signatures can be efficiently linked. Signatures of the same user in different time frames should remain unlinkable though. A stronger version allows public retrieval of the culprits identity without the intervention of a group manager. Because we regard double-spenditure as bad behaviour, we only define a minimum penalty (either detection or identification). We do not require that the damage stops there, it might very well be that double-spenditure in fact results in full traceability of that user’s signatures or even the ability of anyone to sign on that user’s behalf. Needless to say, it is possible to pinpoint the penalty of double expenditure more precisely (note that in the real/ideal-model this is immediate (11)).

With respect to group management, we opted for a slightly less traditional approach. First of all we assume that there is already a legally binding PKI in place with which users can identify but also commit themselves. Secondly we assume that any group public key also implicitly defines the qualified group members under that key. This assumption makes it easier to define security for dynamic groups (either ad hoc or schemes allowing revocation). Even for schemes that claim a constant size group key, such as the one by Ateniese et al. (2) the public key satisfies this property if the transcripts of the Join-protocols are regarded as part of the public key. As noted by Dodis et al. (21) it is not so much the size of the theoretical group public key that matters in reality, but rather the information that is actually needed to sign and verify signatures (assuming that it has already been checked that this information is part of a valid and relevant group public key).

For our definition of list signatures we extend the existing model for group signatures (see (18; 13)) with protocols for detection and identification added and taking into account some recent developments regarding the definition of

group signatures (25; 6; 5) and the discussion above.

A list signature scheme implies various entities: a group manager \mathcal{M} who is responsible for the group's public key and users i who will be list members. The scheme contains protocols for the following tasks:

Key generation which produces the group's public key used to verify signatures, a private key for each group member, and a private key for the group manager. Typically key generation consists of a Setup protocol to initialize the system that will output the secret key of the manager(s) and some related public information; an interactive Join protocol between a user i and the manager after which the user becomes a member of the list, legally bound by his own signature using an existing PKI; a Revoke protocol that the group manager can use to revoke a member; and an Update protocol that group users can use to update their secret key after a change in the public key as a consequence of another user joining or leaving the qualified list of members.

Sign to produce a signature on input of a message, a time frame, the group's public key, and a group member's private key.

Verify which takes as input a message, a time frame, a signature and the group's public key and accepts iff the signature is correct for that message and time frame.

Open which is used by the group manager to prove that a group member did or did not produce a signature.

Rely to determine whether out of a list of signatures based on the same timeframe two (or, in general, k) signatures were produced by the same person (called detection procedure) and, for schemes with identification, by whom.

Note that a scheme with opening but without detection or identification is the known group signature scheme. A scheme with detection or identification but without the capability of opening is called a list signature scheme. A scheme with both opening and detection could be called either a group signature scheme with detection or a list signature scheme with opening. We do not reserve a name for a scheme with neither opening nor detectability possibilities because it is unclear to us how to define them formally.

We consider three security properties. For the second property we make a distinction between group signatures, list signatures and the combination of both. We only describe the properties informally and do not precise the powers of the adversary. In a formalization, the adversary will typically be modelled as a probabilistic polynomial time Turing machine run on input 1^λ that is allowed to introduce honest and corrupt users to the system, corrupt honest players, have honest players revoked, ask honest players for signatures and, if applicable, ask for signatures to be opened.

The only limitations we pose on the adversary are that it can only query one signature per honest player per timeframe and it cannot ask for opening a challenge signature (in the anonymity game). We are also cautious about allowing corruption of the group manager, especially since the group manager can be split according to functionality (e.g., a separate list manager and open manager). In the concrete schemes we will be more concrete about the level of corruption that the scheme can cope with.

Correctness An adversary cannot prevent honest group members from producing valid signatures, nor can the adversary cause the detection of the signatures of an honest group member.

Soundness An adversary can produce at most one valid signature per time frame per corrupted player without being detected, or without the identity of a corrupted group member being released. Note that signatures obtained from querying honest signers do not count as produced by the adversary. (If applicable, an adversary cannot produce a valid signature that does not open to a corrupted player.)

Anonymity Given a set of signatures over different timeframes, the adversary cannot determine whether two of them were signed by the same person.

Note that in the case of identification the second clause of correctness is superfluous. An adversary capable of causing identification of honest users already breaches anonymity. We are not aware of a similar argument for schemes with just detection.

The detectability feature of list signature schemes automatically implies that theoretically list members can disavow or claim a signature without the need to know the randomness that created it, based on the well known result that all of NP can be proven in zero-knowledge. As pointed out by Camenisch ¹, this has the side-effect that a coalition encompassing all users bar one, can prove that a signature originated from the last remaining honest user. In ordinary group signature schemes, this ought not to be possible.

5 Basic Ideas for Realizing List Signature Schemes

In this section we informally describe how to adapt group signature schemes into list signature schemes. We will concentrate on detectability, since getting rid of the opening facility of group signatures, if required, is usually relatively straightforward. In the literature two important types of group signatures exist, those whose efficiency is (largely) independent of the group size and those for which this is not the case, typically resulting in linear dependency.

¹ Private communication, 2000

Interestingly, most large group signature schemes are based on the Strong RSA assumption coupled with (a version of) the Decisional Diffie Hellman assumption. Small group signature schemes can also be easily implemented based on groups in which the DDH problem is assumed hard.

This commonality of different schemes leads us to a generally applicable solution for turning known group signature schemes into list signature schemes. Let us first define some notation and terminology. We let G be a group of possibly unknown order in which the DDH problem is assumed to be hard, even if the group decomposition of G is known. We also assume the existence of a hash function $H : \{0, 1\}^* \rightarrow G$. Furthermore, we observe that in all schemes we are aware of each user has a unique secret key x_i that lives in the exponent group (although not necessarily reduced modulo the group order).

The basic idea for performing detection is simple. Given the description of a timeframe T , the user computes and publishes $H(T)^{x_i}$, along with a NIZK-proof that the same secret key was used for the computation of $H(T)^{x_i}$ and the rest of the signature. Note that most, if not all, known constructions of group signatures already employ a NIZK-proof involving x_i during the signing state and that adapting these proofs can always be done theoretically and quite efficiently in practice (which will become clear from our examples).

It is intuitively clear that an otherwise honest signer who signs twice in the same timeframe will be caught since the values of $H(T)^{x_i}$ will collide in these two signatures. On the other hand, it is extremely unlikely that the signatures of two honest users cause detection. In fact, it will also be hard for an adversary to cause detection based on an honest user's signature and one of his own signatures, since the proof of knowledge that is part of any valid signature requires him to actually know a value x congruent to the secret key x_i of the honest user (modulo the group order). This secret is usually well protected from an adversary.

Linking signatures of the same user, but within different time frames will be difficult. Loosely speaking, the proof of knowledge will not aid an adversary in linking signatures since it is a NIZK-proof. Furthermore, the value $H(T)^{x_i}$ can be regarded to be independent of the rest of the signature if H is modelled as a random oracle. Here we use the fact that the rest of the signature is based on a group signature scheme where time frames are not defined, so H will not be queried on T in that part of the signature. Moreover, if the original group signature scheme was secure, that part of the list signature will not give the signer away. All that remains are the values $H(T)^{x_i}$ for a fixed x_i but various T . Under the random oracle model, the elements output by H are uncorrelated or, more precisely, even for an adversary who can adaptively choose the values T_j the distribution $H(T_j)_{j=1, \dots, n}$ will be indistinguishable from the uniform distribution over G^n , provided all the T_j are different. But then the DDH

assumption will guarantee that the distribution $(H(T_j), H(T_j)^x)_{j=1, \dots, n}$ where x is randomly chosen in the exponent group is computationally indistinguishable from the uniform distribution over G^{2n} , since it is well known that the hardness of the DDH problem based on quadruples implies the hardness of the DDH problem based on any $2n$ -tuple as long as n is polynomial in the security parameter.

The idea of detection using the exponentiation of a random group element to a fixed secret also appears in the context of for instance toggling schemes (31), fair E-cash systems (32; 16), direct anonymous attestation (11) and traceable signatures (24). We note that it can be advantageous to use a different group than G to perform the detection (cf. (11)) to improve the efficiency or to ease the implementation of identification.

Identification A list signature scheme providing detection can also be extended to provide identification of the culprit using techniques based on secret sharing. We will assume that the group G is of large prime order q .

The easiest 2-out-of- q threshold sharing schemes for \mathbb{Z}_q that exists is one based on lines. Let $s \in \mathbb{Z}_q$ be a secret, then the dealer picks a random point $r \in \mathbb{Z}_q$ and hands out shares $(X, r + sX) \in \mathbb{Z}_q^2$, i.e., points on a line. Given one point the slope s is still information theoretically hidden, but clearly two points or shares define the line and allow retrieval of the line.

A list signature scheme can be equipped with the identification functionality by requiring that, as part of his signature, the user releases a point on a line that contains his identity. Obviously a user should not be allowed to use the same point all over again, but this can easily be ensured by using a hash of the message to be signed. A problem though is that signatures in different time frames should not reveal anything, which basically requires a fresh line for each time frame. This technicality can be overcome by exploiting the linearity of the secret sharing scheme and conducting the secret sharing scheme in the exponent group \mathbb{Z}_q of G , handing out shares (X, g^{r+sX}) instead. The secret s will remain the same, but the randomness r will depend both on the time frame and the identity of the user (basically by performing Diffie Hellman key agreement in some sense). Although only g^s can be reconstructed this way, identification of the user is ensured.

6 Small Groups

In this section we describe how the construction above can be used in conjunction with a group signature scheme based on the CDS-technique to perform 1-out-of- N proofs (19) and standard discrete logarithm based tools to form

a list signature with identification capability. The scheme is actually of the *ad-hoc* type, meaning that the list of members belonging to a public key only need to be determined at the time of signing.

Setup The group manager picks a group \mathbb{G}_q of prime order q and publishes (\mathbb{G}_q, q) , together with two randomly chosen generators g and h of \mathbb{G}_q . The group \mathbb{G}_q implicitly defines a hash-function $H : \{0, 1\}^* \rightarrow \mathbb{G}_q$ and a hash-function $H' : \{0, 1\}^* \rightarrow \mathbb{Z}_q$. The Decision Diffie-Hellman problem is assumed to be hard in the group \mathbb{G}_q .

Note that the group manager does not have a secret key and is not involved in the remainder of the list signature scheme.

Join A user can join by picking a random $x \in \mathbb{Z}_q$ and publishing $y = g^x$, together with a proof of knowledge of x . The user also publishes some $z \in \mathbb{Z}_q$. A user's z has to be unique. Henceforth we will assume that user i is legally bound to his public key pair (y_i, z_i) and denote $x_i = \log_g y_i$.

To sign anonymously on behalf of a list of users, the group public key is computed as the concatenation of the public keys of the individual members.

Sign Let $(y_j, z_j)_{j=1, \dots, N}$ be the group public key and let $i \in \{1, \dots, N\}$. User i wishing to sign a message m in time frame T first computes $s = H(T, 1)$, $t = H(T, 0)$, and $X = H'(m, T)$. It then publishes values $T_1 = t^{x_i}$ and $T_2 = s^{x_i} z_i^X$ together with a NIZK-proof that for some $j \in \{1, \dots, n\}$ it holds that the user knows an $x \in \mathbb{Z}_q$ such that $y_j = g^x$, $T_1 = t^x$, and $T_2 z_j^{-X} = s^x$. Here the technique developed in Section 3 can be employed to save some work. The user first proves knowledge of $\log_t T_1$ and then performs a 1-out-of- n proof of the equality of the discrete logarithms $\log_{gt} y_j T_1$ and $\log_s T_2 z_j^{-X}$.

Verify Verify that the public key is correct, i.e., only consists of values resulting from the Join protocol, and verify the NIZK-proof provided by the signature.

Rely If two signatures (T_1, T_2) on message m and (T'_1, T'_2) on message m' based on the same timeframe T satisfy $T_1 = T'_1$ compute $z' = (T_2/T'_2)^{1/(X-X')}$, where $X = H'(m, T)$ and $X' = H'(m', T)$. Identify the user with $z = z'$.

6.1 Security

Our claim is that the scheme just described is secure in the ROM under the DDH assumption. Since the group manager is hardly involved in the scheme, he may also be totally corrupted, provided the key generation performed by him during Setup is guaranteed not to allow him to introduce a somehow weak group \mathbb{G}_q .

Correctness We first remark that we only have to contend ourselves with the first clause, since we are dealing with a scheme with identification. The first clause is trivially satisfied, since an honest user will only ever commit to

a public key (y, z) for which it knows $\log_g y$.

Soundness This follows from the soundness of the zero-knowledge proof used in the signing algorithm and the fact that, under anonymity, the adversary cannot know the secret key of honest users.

Anonymity This boils down to a straightforward reduction to the DDH.

6.2 Efficiency

Just to give a flavour of the efficiency of the scheme above, suppose it is implemented based on an elliptic curve group with $\lg q \approx 160$. In that case a single user's public key consists of two points on the elliptic curve, taking about 320 bits in total (using standard point compression techniques). The public key of a group of N members will be approximately $320N$ bits long. A signature takes 320 bits for publishing T_1 and T_2 plus an additional 320 bits per group member for the proof of knowledge, i.e., a grand total of $320(N + 1)$ bits for a signature.

7 Large Groups

Our proposal is based on the group signature scheme of Ateniese et al. (2). In a nutshell, each user is issued with a triple (A, e, x) such that $A^e = a^x a_0 \pmod n$. The pair (A, e) is public and linked to the identity of the user, the value x is his secret. A signature is basically a proof of knowledge of such a pair, made non-interactively using the Fiat-Shamir heuristic (this is where the message comes in).

The scheme still needs a revocation scheme to be complete. Various revocation principles have been proposed ((3), (10), (15) and (26)), but (3) and (10) are not efficient enough and (26) is broken. There remains the article of Camenisch and Lysyanskaya (15) based on earlier work by Barić and Pfitzmann (4) and that is the one we will use in the following. Basically, a revocation manager publishes u and v such that $v = u^{\prod_i e_i}$ over all group member i . Camenisch and Lysyanskaya fix a value for u and, by changing v , users are added or removed from the qualified list. To determine the qualified list, two sets are maintained, E_{add} and E_{del} . The set of qualified users is precisely those with a value $e \in E_{qual} = E_{add} \setminus E_{del}$ and the public key should at any time satisfy $v = u^{\prod_{e \in E_{qual}} e}$ (if the public key is corrupt, we will by definition assume that no user is qualified).

Clearly qualified users will be able to proof knowledge of an e -th root of v with a corresponding membership certificate simply by computing $B = u^{(\prod_i e_i)/e}$ where $e_i \in E_{qual}$. Camenisch and Lysyanskaya however do not publish u and show that in fact the scheme can be made more efficient: users only need to update their 'root' when other users are added or deleted and this update is linear in the number of modifications, but constant in the number of users that do not change status.

We claim that publishing u is beneficiary for two reasons. First of all it ensures that qualified users always have a membership certificate, even in the face of a corrupt revocation manager. Secondly it allows us to present an alternative that removes the update necessity when users are added. Normally when a user with exponent e is added, the revocation manager computes a new value v by v^e and hence all other users have to raise their secrets by e as well to keep track of the right root of v . However, the revocation manager might equally well update u by $u^{1/e}$ and computing $v^{1/e}$ for the new list member. Since this does not change v , other users do not need to change their respective B 's. Note however that this trick falls outside of the standard dynamic accumulator theory used by Camenisch and Lysyanskaya (15), so we cannot carbon copy their security proof. Moreover, it requires that either \mathcal{M}_R can compute roots, or it already knows a predetermined set of roots for the initial value of u , in which case \mathcal{M}_J must be forced to use these predetermined e 's during the Join-protocol.

We have to introduce some security parameters and refer to the original work by Ateniese et al. (2) for more details. Following Brickell et al. (11), we do provide recommendations for the parameters (in parentheses). We will need $\epsilon > 1$ and integers k (80) and l_p (1024). Moreover, λ_1 (4258), λ_2 (4096), γ_1 (4422) and γ_2 (4260) will denote lengths that define the intervals $\Lambda = (2^{\lambda_1} - 2^{\lambda_2}, 2^{\lambda_1} + 2^{\lambda_2})$ and $\Gamma = (2^{\gamma_1} - 2^{\gamma_2}, 2^{\gamma_1} + 2^{\gamma_2})$. Furthermore, we will need a collision-resistant hash function $H' : \{0, 1\}^* \rightarrow \{0, 1\}^k$.

7.1 Setup Protocol

Setup is run by the group manager in two different incarnations: one called \mathcal{M}_J running the join protocol and one responsible for revocation \mathcal{M}_R . The join manager is the most important one in the sense that \mathcal{M}_J should remain uncorrupted to achieve soundness, whereas \mathcal{M}_R may be corrupted.

- (1) Manager \mathcal{M}_J selects distinct primes p' and q' of size l_p , such that $p = 2p' + 1$ and $q = 2q' + 1$ are primes. It computes and publishes the modulus $n = pq$ together with a proof that n is a safe RSA-modulus (1). It keeps

the factorisation of n as secret key.

We assume that publishing n implicitly defines a hash function $H : \{0, 1\}^* \rightarrow QR(n)$. In the random oracle model, H' can be assumed to produce a distribution statistically close to the uniform one of generators of $QR(n)$ and that different calls to H' produce independent and hence uncorrelated outcomes unless the calls are on the same input. In particular, nothing will ‘leak’ about respective discrete logarithms, not even to \mathcal{M}_J who knows the factorisation of n .

- (2) Manager \mathcal{M}_J chooses random elements a, a_0, g and h in $QR(n)$ and publishes them. It sets up (empty for now) a database DB.
- (3) Manager \mathcal{M}_R chooses random elements u, f in $QR(n)$, sets $v = u$ and publishes u, v and f . It sets up (empty for now) E_{add} and E_{del} . For anonymity against \mathcal{M}_R it is essential that f is uncorrelated to (g, h) .

At the end of this protocol, we denote the list public key by $pk = (n, a, a_0, g, h, u, v, f)$ together with DB, E_{add} , E_{del} , and denote \mathcal{M}_J 's private key by $sk_{\mathcal{M}} = p'$. Note that \mathcal{M}_R does not have a private key specific to the group signature scheme, although it is implicitly understood to have a legally binding key outside the group signature scheme in order to sign its part of the group public key (u, v) .

7.2 Join Protocol

This algorithm is based on the Join-protocol that is part of the group signature scheme by Ateniese et al. (2).

- (1) User i and manager \mathcal{M}_J engage in a two-party protocol, at the end of which the user knows $x \in \Lambda$ and the manager knows $a^x \bmod n$. The protocol is such that the user cannot influence the choice (or distribution) of x and the manager learns nothing beyond $a^x \bmod n$ (this can be formalized (24)).
- (2) The manager picks a prime $e \in \Gamma$ not yet in DB and computes $A = (a^x a_0)^{1/e} \bmod n$. It sends (A, e) to the user.
- (3) The user checks that neither A nor e already occurs in the database, that e is a prime in Γ and that $A^e = a^x a_0 \bmod n$. If so, it returns (A, e) together with a legally binding signature on it incorporating (a, a_0, n) .
- (4) Manager \mathcal{M}_J adds (A, e) to the database, together with i 's identity and signature.
- (5) Manager \mathcal{M}_R checks that e is a prime in Γ that occurs only once in the database. If so, it adds e to E_{add} , sets $B = v$, sends B to the user, and updates the public key by $v = v^e \bmod n$. In our variation \mathcal{M}_R leaves v alone, updates $u = u^{1/e}$ and sends $B = v^{1/e}$ to the user.

During this protocol, a new list member i will obtain from the managers a private key $sk_i = x$ and a membership certificate (A, e, B) such that $A^e = a^x a_0 \bmod n$ and $B^e = v \bmod n$. Note that the membership certificate is public: (A, e) is part of the database and B can be recomputed based on u, v and the sets E_{add} and E_{del} .

7.3 Revoke Protocol

In case of a revocation of the list member i with membership certificate (A_i, e_i, B_i) , manager \mathcal{M}_R has to compute a new value for v being $v^{1/e_i} \bmod n$. He then modifies adds e_i to E_{del} and publishes the new v and E_{del} .

Although it seems revocation requires an e_i 'th root computation on \mathcal{M}_R 's behalf, involving the factorisation of n , the group manager can actually recompute v based on the fixed element u and the sets E_{add} and E_{del} . Note that the workload for the group manager for revoking one group member is linear in the number of participants this way. If \mathcal{M}_R does know the factorisation of n , it can obviously perform the root computation directly and independently of the number of participants.

If it is undesired to give \mathcal{M}_R access to the secret key of \mathcal{M}_J , an efficient solution is the introduction of a second modulus n' of which only \mathcal{M}_R knows the factorisation. In this case u, v , and f will be elements of $QR(n')$. Using a different group for the revocation manager only marginally complicates the proofs needed in the Sign protocol (the main tool is a proof of knowledge of a discrete logarithm with respect to different modulus, see e.g. (8; 7)).

7.4 Update Protocol

After a Registration protocol, a list member i (using E_{add}) has to replace B_i in his membership certificate by $B_i^e \bmod n$, where e is a new entry of E_{add} . However, if our variation is used where \mathcal{M}_R updates u instead of v , old list members do not have to update their key when new members join.

After a Revocation protocol revoking user i , a list member $j \neq i$ (using E_{del}) has to replace the value B_j in his certificate by $B_j^b v^a \bmod n$ where a and b are such that $ae_j + be_i = 1$ and where e_i is a new entry of E_{del} .

In either case, the user j always knows a couple (e_j, B_j) such that $B_j^{e_j} = v \bmod n$.

7.5 Sign Protocol

In the original scheme of Ateniese et al., the signer first publishes $T_1 = A_i g^w \bmod n$, $T_2 = h^w \bmod n$, and $T_3 = g^{e_i} h^w \bmod n$, where w is a randomly chosen value, followed by a proof of knowledge of e_i, x_i, w , and $(e_i w)$ such that $T_1^e = a^x a_0 g^{(e_i w)}$, $T_2 = h^w$, $T_2^{e_i} = h^{(e_i w)}$, and $T_3 = g^{e_i} h^w$, all modulo n . The proof of knowledge includes range checks on x and e , but does not involve the primality of e . The main tool is a proof of knowledge of a discrete logarithm in a group of unknown order (see e.g. (14; 8; 7; 24)).

Our first observation is that publishing T_3 and explicitly proving knowledge of w is actually superfluous in the above scheme (or, for that matter, in a whole range of schemes derived from it). Our second observation is that T_1 and T_2 both seem necessary, even if opening of the signature based on knowledge of $\alpha = \log_g h$ is not required. Our third observation is that publishing $H(T)^x \bmod n$ will allow detectability, where T is the time frame. Finally, we remark that including Bf^w in the signature will show that the user is qualified (cf. (15)).

This leads us to the following signing algorithm for message m , time frame T and based on membership certificate (A, e, x, B) .

- (1) The signer picks w at random modulo n , and computes

$$\begin{aligned} T_1 &= Ag^w \bmod n \\ T_2 &= h^w \bmod n \\ T_3 &= Bf^w \bmod n \\ T_4 &= t^x \bmod n \end{aligned}$$

- (2) The signer proves knowledge of $e \in \Gamma, x \in \Lambda$, and existence of (ew) such that

$$\begin{aligned} T_1^e &= a^x a_0 g^{(ew)} \bmod n \\ T_2^e &= h^{(ew)} \bmod n \\ T_3^e &= v f^{(ew)} \bmod n \\ T_4 &= t^x \bmod n \end{aligned}$$

where the message to be signed is hashed into the challenge.

The proof of knowledge is fairly standard and takes just over three elements, each slightly bigger than n^2 .

7.6 Verify and Rely Protocols

A verifier can check the validity of a signature by simply verifying the proof of knowledge.

Moreover, everyone who has access to all signatures for a particular time frame can easily see if a user has signed twice or more by observing the value of T_4 . The user cannot cheat (by using another value) because T_4 is linked with T_1 by the proof of knowledge and the private key x

7.7 Security of the Proposed List Signature Scheme

In this section, we examine the security of our list signature scheme. Once again, since we did not provide a formal model of security, we suffice with an informal argumentation why the list signature scheme just presented is secure.

Correctness The first clause of correctness is satisfied because an honest user i will only sign a pair (A, e) if he knows an x such that (A, e, x) is a valid certificate. Moreover, a user will only be classified as qualified if $e \in E_{qual}$ and $v = u^{\prod_{e \in E_{qual}} e}$, from which B satisfying $B^e = v$ can be efficiently reconstructed. Hence any honest qualified will have the knowledge to pass through the NIZK proof required for a signature (where we use completeness of this proof).

The second clause of correctness is also satisfied, since the only way an adversary can cause detection is by posting a signature for the same time frame with t^x , including a proof of knowledge of a value x' such that $x' = x \pmod{pq'}$ where x is the user's secret key. But if the adversary knows such an x' , he can also identify signatures from that user, breaching anonymity.

Soundness We claim that the list signature scheme is sound against attacks excluding the Join manager. Note that the \mathcal{M}_J can always introduce ghost members that have valid certificates but that do not appear in the database. This holds even if the revocation manager is not corrupted and uses a different group of order unknown to \mathcal{M}_J , since the ghost users can piggyback on the e 's of honest users. Soundness is based on the following argumentation.

- (1) Without loss of generality we can assume that the adversary knows a value $\alpha = \log_h g$ and a value $\beta = \log_h f$, since knowledge of these values only adds strength to the adversary.
- (2) If an adversary posts a valid signature (T_1, T_2, T_3, T_4) , he also proves, by virtue of the soundness of the zero-knowledge proof in the signature, that he knows $e \in \Gamma$ and $x \in \Lambda$ for which an (ew) exists such that

$T_1^e = a^x a_0 g^{(ew)}$, $T_2^e = h^{(ew)}$, and $T_3^e = v f^{(ew)}$. But then $(T_2^{-\alpha} T_1)^e = a^x a_0$ and $(T_2^{-\beta} T_3)^e = v$. Since we assume the adversary knows α and β , it can compute a valid membership certificate (A, e, x, B) based on any valid signature it issues. Note that the random oracle model is used for the soundness of the NIZK-proof.

- (3) By virtue of the proof of equality of x in T_4 and x in the membership certificate, each certificate can be used only once within the same timeframe without triggering detection.
- (4) Under the strong RSA assumption, the Join-protocol cannot be used to obtain more certificates than queried for (2, Theorem 1).
- (5) Under anonymity of the users (see below), the value x of honest users is unknown for the adversary, hence certificates of honest users are of no avail.
- (6) One cannot piggyback on the activation of an honest user because of the security (under the strong RSA assumption) of the dynamic accumulator used for list management (15).

Anonymity First of all, if the Join-protocol is correctly implemented, it will only leak $a^x \bmod n$ (some care has to be taken to shield the protocol against concurrent attacks).

In the random oracle model the proof of knowledge that is part of the signature can be proven statistically zero-knowledge (note that concurrency is not an issue, since a signature is a one-round protocol). Hence all that leaks from a signature is the quadruple (Ag^w, h^w, Bf^w, t^x) . An adversary that could distinguish anything about A and/or B would brake the semantic security of ElGamal encryption, which is hard under the Decisional Diffie Hellman assumption. Hence a signature essentially only leaks t^x . Luckily the problem of distinguishing the distribution $(a, t_1, t_2, \dots, t_l, a^x, t_1, \dots, t_l^x)$ with a and t_1, \dots, t_l uniformly random in $QR(n)$ and x in Γ from the one of $2l + 2$ uniformly distributed elements in $QR(n)$ reduces to standard DDH.

7.8 Efficiency

The group's public key consists of the modulus n and six elements of $QR(n)$, taking in total 7168 bits. A signature involves the posting of four elements in $QR(n)$ together with a proof of knowledge of about 3 elements slightly bigger than n^2 . In total a signature can be expected to take about 23250 bits. Hence for the schemes presented in this work, at around 70 participants the large group scheme becomes more efficient than the one based on small groups and the CDS-technique.

We would like to point out though that recently more efficient group signature

schemes have appeared for large groups, that can also be adapted to list signature schemes. In fact, the direct anonymous attestation scheme by Brickell et al. (11) is already very close to a list signature scheme if the TPM and the host in their scheme are regarded as a single user. On the downside we would like to note that presently all large group signature schemes seem to be based on the Strong RSA assumption (and the DDH), whereas the small group schemes can be based on just a group in which the DDH is hard, allowing groups over elliptic curves. Given the current track record of attacking the respective problems (27), it is to be expected that the moduli for the large group schemes have to grow faster than the group sizes of the elliptic curves, thus the break-even point can be expected to shift preferring small group and list signature schemes.

8 Application to Electronic Voting

Many proposals for electronic voting schemes use blind signatures as a building block. In such schemes, each voter is issued a single blind signature for use in a particular election. During the election, the voter uses its blind signature to authenticate a vote, which is submitted via an anonymous channel. The election result is determined by collecting all submitted votes and accompanying blind signatures. The security of these voting schemes relies on the fact that blind signatures cannot be forged, accepting each blind signature at most once to prevent voters from voting multiple times. Finally, ballot secrecy is achieved since the blind signatures used to authenticate a vote do not reveal any information on the identity of the voter who submitted it.

We note that list signatures can be used as a substitute for blind signatures in such voting schemes. Instead of authenticating a vote by means of a blind signature, a list signature is used, preserving the security and privacy properties of the voting scheme. A first advantage of the use of list signatures is that there is one less interaction for the voter during the election process. Another potential advantage is that the computational complexity and storage complexity improves for the voters. In a list signature scheme, the storage for a user is $O(1)$ and the work for generating a signature is also $O(1)$ (assuming a scheme for large groups). Also, there is no practical limit on the number of signature that can be produced by a user.

When using blind signatures, a user must decide beforehand how many blind signatures it wishes to produce. There are techniques to limit the storage and work when handling batches of blind signatures. Such techniques were introduced by Chaum (at DigiCash), and later investigated by Fiat (22) and Schoenmakers (30). These ideas can be extended to 'packing' many blind signatures, but achieving both $O(1)$ storage and $O(1)$ work may require addi-

tional assumptions.

References

- [1] J. Algesheimer, J. Camenisch, and V. Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In Yung (33), pages 417–432.
- [2] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *Advances in Cryptography—Crypto’00*, volume 1880 of *Lecture Notes in Computer Science*, pages 11–15. Springer-Verlag, 2000.
- [3] G. Ateniese, D. Song, and G. Tsudik. Quasi-efficient revocation of group signatures. In M. Blaze, editor, *FC’02*, volume 2357 of *Lecture Notes in Computer Science*, pages 183–197. Springer-Verlag, 2003.
- [4] N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Fumy (23), pages 480–484.
- [5] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, *Advances in Cryptography—Eurocrypt’03*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer-Verlag, 2003.
- [6] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. Technical Report 77, IACR’s ePrint Archive, 2004.
- [7] F. Boudot. On the soundness of Girault’s scheme. Poster paper of Eurocrypt 2000. Rump Session, 2000.
- [8] F. Boudot and J. Traoré. Efficient publicly verifiable secret sharing schemes with fast or delayed recovery. In V. Varadharajan and Y. Mu, editors, *ICICS’99*, volume 1726 of *Lecture Notes in Computer Science*, pages 87–102. Springer-Verlag, 1999.
- [9] C. Boyd, editor. *Advances in Cryptography—Asiacrypt’01*, volume 2248 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.
- [10] E. Bresson and J. Stern. Efficient revocation in group signatures. In K. Kim, editor, *PKC’01*, volume 1992 of *Lecture Notes in Computer Science*, pages 190–206. Springer-Verlag, 2001.
- [11] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation, 2004.
- [12] C. Cachin and J. Camenisch, editors. *Advances in Cryptography—Eurocrypt’04*, volume 3027 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.
- [13] J. Camenisch. Efficient and generalized group signatures. In Fumy (23), pages 465–479.
- [14] J. Camenisch. *Group Signature Schemes and Payment Systems Based on*

- the Discrete Logarithm Problem*. PhD thesis, ETH Zurich, 1998.
- [15] J. Camenisch and A. Lysyanskaya. Efficient revocation of anonymous group membership certificates and anonymous credentials. In Yung (33), pages 93–118.
 - [16] S. Canard and J. Traoré. On fair e-cash systems based on group signature schemes. In R. Safavi-Naini and J. Seberry, editors, *ACISP'03*, volume 2727 of *Lecture Notes in Computer Science*, pages 237–248. Springer-Verlag, 2003.
 - [17] D. Chaum and T. Pedersen. Wallet databases with observers. In E. Brickell, editor, *Advances in Cryptography—Crypto'92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer-Verlag, 1993.
 - [18] D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *Advances in Cryptography—Eurocrypt'91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer-Verlag, 1991.
 - [19] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. Desmedt, editor, *Advances in Cryptography—Crypto'94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer-Verlag, 1994.
 - [20] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Fumy (23), pages 103–118.
 - [21] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In Cachin and Camenisch (12), pages 609–626.
 - [22] A. Fiat. Batch RSA. *Journal of Cryptology*, 10(2):75–88, 1997.
 - [23] W. Fumy, editor. *Advances in Cryptography—Eurocrypt'97*, volume 1233 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.
 - [24] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In Cachin and Camenisch (12), pages 571–589.
 - [25] A. Kiayias and M. Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor holders. Technical Report 76, IACR's ePrint Archive, 2004.
 - [26] H. Kim, J. Lim, and D. Lee. Efficient and secure member deletion in group signature schemes. In D. Won, editor, *ICISC'00*, volume 2015 of *Lecture Notes in Computer Science*, pages 150–161. Springer-Verlag, 2000.
 - [27] A. K. Lenstra. Unbelievable security: matching AES security using public key systems. In Boyd (9), pages 67–86.
 - [28] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In Boyd (9), pages 552–565.
 - [29] C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.
 - [30] B. Schoenmakers. Basic security of the ecash payment system. In B. Preneel and V. Rijmen, editors, *State of the Art in Applied Cryptography*, volume 1528 of *Lecture Notes in Computer Science*, pages 338–352. Springer-Verlag, 1997. Correct version available at <http://www.win.tue.nl/~berry/papers/cosic.pdf>.
 - [31] M. Stam. Toggling schemes for electronic voting. Master's thesis, Tech-

nische Universiteit Eindhoven, 1999.

- [32] J. Traoré. Group Signatures and Their Relevance to Privacy-Protecting Off-Line Electronic Cash Systems. ACISP'99, volume 1587 of LNCS, pages 228-243. Springer-Verlag, 1999.
- [33] M. Yung, editor. *Advances in Cryptography—Crypto'02*, volume 2442 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.