Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials

Jan Camenisch
IBM Research
Zurich Research Laboratory
CH–8803 Rüschlikon
jca@zurich.ibm.com

Anna Lysyanskaya
MIT LCS
200 Technology Square
Cambridge, MA 02139 USA
anna@theory.lcs.mit.edu

February 11, 2002

Abstract

An accumulator scheme, as introduced by Benaloh and de Mare [BdM94] and further studied by Barić and Pfitzmann [BP97], is an algorithm that allows one to hash a large set of inputs into one short value, called the *accumulator*, such that there is a (short) witness that a given input was incorporated into the accumulator. At the same time, it is infeasible to find a witness for a value that was not accumulated.

We put forward the notion of a *dynamic accumulator*, which is an accumulator that allows one to dynamically add and delete inputs, such that the cost of an add or delete is independent of the number of accumulated values. We achieve this under the strong RSA assumption. For this construction, we also show an efficient zero-knowledge protocol for proving that a committed value is in the accumulator.

Dynamic accumulators enable efficient membership revocation in the anonymous setting. Our construction is especially suitable for membership revocation in group signature and identity escrow schemes, such as the one due to Ateniese et al. [ACJT00], and efficient revocation of credentials in anonymous credential systems, such as the one due to Camenisch and Lysyanskaya [CL01a]. Applying our method to these schemes enables membership revocation and yet does not significantly increase the complexity of any of the operations. In particular, the cost of a membership verification or credential showing increases by only a small constant factor, less than 2. All previously known methods (such as the ones due to Bresson and Stern [BS01] and Ateniese and Tsudik [AT01]) incur an increase in these costs that is linear in the number of members.

Keywords. Dynamic accumulators, anonymity, certificate revocation, group signatures, credential systems, identity escrow.

1 Introduction

Suppose a set of users is granted access to a resource. This set changes over time: some users are added, and for some, the access to the resource is revoked. When a user is trying to access the resource, some verifier must check that the user is in this set. The immediate solution is to have the verifier look up the user in some database to make sure that the user is still allowed access to the resource in question. This solution is expensive in terms of communication. Another approach is of certificate revocation chains, where every day eligible users get a fresh certificate of eligibility. This is somewhat better because the communication burden is now shifted from the verifier to the user, but still suffers the drawback of high communication costs, as well as the computation costs needed to reissue certificates. Moreover, it disallows revocation at arbitrary time as need arises. A satisfactory solution to this problem has been an interesting question for some time, especially in a situation where the users in the system are anonymous.

Accumulators were introduced by Benaloh and de Mare [BdM94] as a way to combine a set of values into one short accumulator, such that there is a short witness that a given value was incorporated into the accumulator. At the same time, it is infeasible to find a witness for a value that was not accumulated. Extending the ideas due to Benaloh and de Mare [BdM94], Barić and Pfitzmann [BP97] give an efficient construction of so-called collision-resistant accumulators, based on the strong RSA assumption.

We propose a variant of the cited construction with the additional advantage that, using additional trapdoor information, the work of deleting a value from an accumulator can be made independent of the number of accumulated values, at unit cost. Better still, once the accumulator is updated, updating the witness that a given value is in the accumulator (provided that this value has not been revoked, of course!) can be done without the trapdoor information at unit cost. Accumulators with these properties are called *dynamic*. Dynamic accumulators are attractive for the application of granting and revoking privileges.

In the anonymous access setting, where a user can prove eligibility without revealing his identity, revocation appeared impossible to achieve, because if a verifier can tell whether a user is eligible or ineligible, he seems to gain some information about the user's identity. However, it turns out that this intuition was wrong! Indeed, using accumulators in combination with zero-knowledge proofs allows one to prove that a committed value is in the accumulator. We show that this can be done efficiently (i.e., *not* by reducing to an NP-complete problem and then using the fact that $\mathsf{NP} \subseteq \mathsf{ZK}$ [GMW87] and *not* by using cut-and-choose for the Barić and Pfitzmann's [BP97] construction).

From the above, we obtain an efficient mechanism for revoking group membership for the Ateniese et al. identity escrow/group signature scheme [ACJT00] (the most efficient secure identity escrow/group signature scheme known to date) and a credential revocation mechanism for Camenisch and Lysyanskaya's [CL01a] credential system. The construction can be applied to other such schemes as well. The idea is to incorporate the public key for an accumulator scheme into the group manager's (resp., organization's) public key, and the secret trapdoor of the accumulator scheme into the corresponding secret key. Each time a user joins the group (resp., obtains a credential), the group manager (resp., organization) gives her a membership certificate (resp., credential certificate). An integral part of this certificate is a prime number e. This will be the value added to the accumulator when the user is added, and deleted from the accumulator if the user's privileges have to be revoked. This provably secure mechanism does not add any significant communication or computation overhead to the underlying schemes (at most a factor of 2). We note that both our dynamic accumulator scheme and the ACJT identity escrow/group signature scheme rely on the strong RSA assumption. While one could add membership revocation using our dynamic accumulator also to other group signature and identity escrow schemes, such a combination would not make much sense as one would get a less efficient scheme and might even require additional cryptographic assumption. We therefore do not discuss the detail involved here.

1.1 Related Work

For the class of group signature schemes [CP95, Cam97] where the group's public key contains a list of the public keys of all the group members, excluding a member is straightforward: the group manager only needs to remove the affected member's key from the list. These schemes, however, have the drawback that the complexity of proving and verifying membership is linear in the number of current members and therefore becomes inefficient for large groups. This drawback is overcome by schemes where the size of the group's public key as well as the complexity of proving and verifying membership is independent of the number of members [CS97, KP98, CM99, ACJT00]. The idea underlying these schemes is that the group public key contains the group manager's public key of a suitable signature scheme. To become a group member, a user chooses a membership public key which the group manager signs. Thus, to prove membership, a user has to prove possession of membership public key, of the corresponding secret key and of a group manager's signature on a membership public key.

The problem of excluding group members within such a framework without incurring big costs has been considered, but until now no solution was satisfactory. One approach is to change the group's public key and reissue all the membership certificates (cf. [AT01]). Clearly, this puts quite a burden on the group manager, especially for large groups. Another approach is to incorporate a list of revoked certificates and their corresponding membership keys into the group's public key [BS01]. In this solution, when proving membership, a user has to prove that his or her membership public key does not appear on the list. Hence, the size of the public key as well as the complexity of proving and verifying signatures are linear in the number of excluded members. In particular, this means that the size of a group signature grows with the number of excluded members.

Song [Son01] presents an alternative approach in conjunction with a construction that yields forward secure group signature schemes based on the ACJT group signature scheme [ACJT00]. While here the size of a group signature is independent of the number of excluded members, the verification task remains computationally intensive, and is linear in the number of excluded group members. Moreover, her approach does not work for ordinary group signature schemes as it relies heavily on the different time periods peculiar to forward secure signatures. Ateniese and Tsudik [AT01] adapt this approach to the ACJT group signature/identity escrow scheme. Their solution retains the property that the verification task is linear in the number of excluded group members. Moreover, it uses so-called double discrete logarithms which results in the complexity of proving/signing and verifying to be rather high compared to underlying scheme (about a factor of 90 for reasonable security parameters).

Finally, we point out that the proposal by Kim et al. [KLL01] is broken, i.e., excluded group members can still prove membership (after the group manager changed the group's key, excluded members can update their membership information in the very same way as non-excluded members).

Thus, until now, all schemes have a linear dependency either on the number of current members, or on the total number of deleted members. As we have noted above, this linear dependency comes in three flavors: (1) the burden being on the group manager to re-issue certificates in every time period; (2) the burden being on the group member to prove that his certificate is different from any of those that have been revoked and on the verifier to check this; or (3) the burden being on the verifier to perform a computational test on the message received from the user for each item in the list of revoked certificates.

In contrast, in our solution no operation is linearly dependent on the number of current or total deleted members. Its only overhead over a scheme without revocation is the following:

We require some public archive that stores information on added and deleted users. Then, the public key (whose size depends only on the security parameter) needs to be updated each time a user is added or deleted. Each user must read the public key from time to time (e.g., prior to proving his membership), and if the public key has changed since the last time he looked, he must read the news in the public archive and then perform a local computation. The amount of data to read and the local computation are linear in the number of changes that have taken place in the meantime, but *not* in the total number of changes. The additional burden on the verifier is simply that he should look at the public key frequently (which seems unavoidable); the verifier need not read the archive.

2 Preliminaries

Let $A(\cdot)$ be an algorithm. By $y \leftarrow A(x)$ we denote that y was obtained by running A on input x. In case A is deterministic, then this y is unique; if A is probabilistic, then y is a random variable.

Let A and B be interactive Turing machines. By $(a \leftarrow A(\cdot) \leftrightarrow B(\cdot) \rightarrow b)$, we denote that a and b are random variables that correspond to the outputs of A and B as a result of their joint computation.

Let b be a boolean function. By $(y \leftarrow A(x) : b(y))$ we denote the event that $b(y)$ is true after y was generated by running A on input x. The statement

$$\Pr[x_1 \leftarrow A_1(y_1);\ x_2 \leftarrow A_2(y_2);\ \ldots\ ; x_n \leftarrow A_n(y_n)\ :\ b(x_n)] = \alpha$$

means that the probability that $b(x_n)$ is TRUE after the value x_n was obtained by running algorithms $A_1, \ldots, A_n$ on inputs $y_1, \ldots, y_n$, is α.

We say that $\nu(k)$ is a negligible function, if for all polynomials $p(k)$, for all sufficiently large k, $\nu(k) < 1/p(k)$.

Let a be a real number. We denote by $\lfloor a \rfloor$ the largest integer $b \leq a$, by $\lceil a \rceil$ the smallest integer $b \geq a$, and by $\lceil a \rfloor$ the largest integer $b \leq a + 1/2$. Let q be a positive integer. Sometime we need to do modular arithmetic centered around 0; in these cases we use 'rem' as the operator for modular reduction rather than 'mod', i.e., $(c \text{ rem } q) = c - \lceil c/q \rfloor q$.

The *flexible* RSA problem is the following. Given an RSA modulus n and a random element $v \in \mathbb{Z}_n^*$ find $e > 1$ and u such that $z = u^e$. The *strong* RSA assumption states that this problem is hard to solve. The strong RSA assumption [BP97, FO97] is a common number-theoretic assumption that, in particular, is the basis for several cryptographic schemes (e.g., [ACJT00, CM98, CS98, GHR99]). By QR_n we denote the group of quadratic residues modulo n.

We use notation introduced by Camenisch and Stadler [CS97] for the various zero-knowledge proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

$$PK\{(\alpha, \beta, \gamma) : y = g^{\alpha}h^{\beta}\ \wedge\ \mathfrak{y} = \mathfrak{g}^{\alpha}\mathfrak{h}^{\gamma}\ \wedge\ (u \leq \alpha \leq v)\}$$

denotes a "*zero-knowledge* P*roof of* K*nowledge of integers* α, β, *and* γ *such that* $y = g^{\alpha}h^{\beta}$ *and* $\mathfrak{y} = \mathfrak{g}^{\alpha}\mathfrak{h}^{\gamma}$ *holds, where* $v < \alpha < u$," where $y, g, h, \mathfrak{y}, \mathfrak{g}$, and $\mathfrak{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\mathfrak{G} = \langle \mathfrak{g} \rangle = \langle \mathfrak{h} \rangle$. The convention is Greek letters denote quantities the knowledge of which is being proved, while all other parameters are known to the verifier. Using this notation, a proof-protocol can be described by just pointing out its aim while hiding all details.

3 Dynamic Accumulators

3.1 Definition

Definition 1. *A secure accumulator for a family of inputs $\{\mathcal{X}_k\}$ is a family of families of functions $\mathcal{G} = \{\mathcal{F}_k\}$ with the following properties:*

Efficient generation: *There is an efficient probabilistic algorithm G that on input 1^k produces a random element f of $\mathcal{F}_k$. Moreover, along with f, G also outputs some auxiliary information about f, denoted aux_f.*

Efficient evaluation: *$f \in \mathcal{F}_k$ is a polynomial-size circuit that, on input $(u, x) \in \mathcal{U}_f \times \mathcal{X}_k$, outputs a value $v \in \mathcal{U}_f$, where $\mathcal{U}_f$ is an efficiently-samplable input domain for the function f; and $\mathcal{X}_k$ is the intended input domain whose elements are to be accumulated.*

Quasi-commutative: *For all k, for all $f \in \mathcal{F}_k$, for all $u \in \mathcal{U}_f$, for all $x_1, x_2 \in \mathcal{X}_k$, $f(f(u, x_1), x_2) = f(f(u, x_2), x_1)$. If $X = \{x_1, \ldots, x_m\} \subset \mathcal{X}_k$, then by $f(u, X)$ we denote $f(f(\ldots(u, x_1), \ldots), x_m)$.*

Witnesses: *Let $v \in \mathcal{U}_f$ and $x \in \mathcal{X}_k$. A value $w \in U_f$ is called a* witness *for x in v under f if $v = f(w, x)$.*

Security: *Let $\mathcal{U}'_f \times \mathcal{X}'_k$ denote the domains for which the computational procedure for function $f \in \mathcal{F}_k$ is defined (thus $U_f \subseteq \mathcal{U}'_f$, $\mathcal{X}_k \subseteq \mathcal{X}'_k$). For all probabilistic polynomial-time adversaries $\mathcal{A}_k$,*

$$\begin{aligned}\Pr[f \leftarrow G(1^k); u \leftarrow \mathcal{U}_f; (x, w, X) \leftarrow \mathcal{A}_k(f, U_f, u) :& \\ X \subset \mathcal{X}_k; w \in \mathcal{U}'_f; x \in \mathcal{X}'_k; x \notin X; f(w, x) = f(u, X)] &= \mathrm{neg}(k)\end{aligned}$$

Note that only the legitimate accumulated values, $(x_1, \ldots, x_m)$, must belong to $\mathcal{X}_k$; the forged value x can belong to a possibly larger set $\mathcal{X}'_k$.

(This definition is essentially the one of Barić and Pfitzmann, with the difference that they do not require that the accumulator be quasi-commutative; as a consequence they need to introduce two further algorithms, one for generation and one for verification of a witness value.)

The above definition is seemingly tailored for a static use of the accumulator. In this paper, however, we are interested in a dynamic use where there is a manager controlling the accumulator, and several users. First, let us show that dynamic addition of a value is done at unit cost in this setting.

Lemma 1. *Let $f \in \mathcal{F}_k$. Let $v = f(u, X)$ be the accumulator so far. Let $v' = f(v, x') = f(u, X')$ be the value of the accumulator when x' is added to the accumulated set, $X' = X \cup \{x'\}$. Let $x \in X$ and w be the witness for x in v. The computation of w' which is the witness for x in v', is independent on the size of X.*

Proof. w' is computed as follows: $w' = f(w, x')$. Let us show correctness using the quasi-commutative property: $f(w', x) = f(f(w, x'), x) = f(f(w, x), x') = f(v, x') = v'$. □

We must also be able to handle dynamic deletions of a value from the accumulator. It is clear how to do that using computations that are linear in the size of the accumulated set X. Here, we restrict the definition so as to make the complexity of this operation independent of the size of X:

Definition 2. *A secure accumulator is* dynamic *if it has the following property:*

Efficient deletion: *there exist efficient algorithms* D, W *such that, if* $\nu = \mathsf{f}(\mathsf{u}, \mathsf{X})$, $\mathsf{x}, \mathsf{x}' \in \mathsf{X}$, *and* $\mathsf{f}(w, \mathsf{x}) = \nu$, *then (1)* $\mathsf{D}(aux_\mathsf{f}, \nu, \mathsf{x}') = \nu'$ *such that* $\nu' = \mathsf{f}(\mathsf{u}, \mathsf{X} \setminus \{\mathsf{x}'\})$*; and (2)* $\mathsf{W}(\mathsf{f}, \nu, \nu', \mathsf{x}, \mathsf{x}') = w'$ *such that* $\mathsf{f}(w', \mathsf{x}) = \nu'$.

Now, we show that a dynamic accumulator is secure against an adaptive adversary, in the following scenario: An accumulator manager sets up the function f and the value u and hides the trapdoor information aux_f. The adversary adaptively modifies the set X. When a value is added to it, the manager updates the accumulator value accordingly. When a value $\mathsf{x} \in \mathsf{X}$ is deleted, the manager algorithm D and publishes the result. In the end, the adversary attempts to produce a witness that $\mathsf{x}' \notin \mathsf{X}$ is in the current accumulator ν.

Theorem 2. *Let* $\mathcal{G}$ *be a dynamic accumulator algorithm. Let* M *be an interactive Turing machine set up as follows: It receives input* $(\mathsf{f}, aux_\mathsf{f}, \mathsf{u})$, *where* $\mathsf{f} \in \mathcal{F}_\mathsf{k}$ *and* $\mathsf{u} \in \mathcal{U}_\mathsf{f}$. *It maintains a list of values* X *which is initially empty, and the current accumulator value,* ν, *which is initially* u. *It responds to two types of messages: in response to the ("ADD",* x*) message, it checks that* $\mathsf{x} \in \mathcal{X}_\mathsf{k}$, *and if so, adds* x *to the list* X *and modifies* ν *by evaluating* f, *it then sends back this updated value; similarly, in response to the ("DELETE",* x*) message, it checks that* $\mathsf{x} \in \mathsf{X}$, *and if so, deletes it from the list and updates* ν *by running* D *and sends back the updated value. In the end of the computation,* M *outputs the current values for* X *and* ν. *Let* $\mathcal{U}'_\mathsf{f} \times \mathcal{X}'_\mathsf{k}$ *denote the domains for which the computational procedure for function* $\mathsf{f} \in \mathcal{F}_\mathsf{k}$ *is defined. For all probabilistic polynomial-time adversaries* $\mathcal{A}_\mathsf{k}$,

$$\Pr[((\mathsf{f}, aux_\mathsf{f}) \leftarrow \mathsf{G}(1^\mathsf{k}); \mathsf{u} \leftarrow \mathcal{U}_\mathsf{f}; (\mathsf{x}, w) \leftarrow \mathcal{A}_\mathsf{k}(\mathsf{f}, \mathsf{U}_\mathsf{f}, \mathsf{u}) \leftrightarrow \mathsf{M}(\mathsf{f}, aux_\mathsf{f}, \mathsf{u}) \rightarrow (\mathsf{X}, \nu) : \\ w \in \mathcal{U}'_\mathsf{f}; \mathsf{x} \in \mathcal{X}'_\mathsf{k}; \mathsf{x} \notin \mathsf{X}; \mathsf{f}(w, \mathsf{x}) = \mathsf{f}(\mathsf{u}, \mathsf{X})] = \operatorname{neg}(\mathsf{k})$$

Proof. Let us exhibit a reduction from the adversary that violates the theorem to the adversary that breaks the collision-resistance property of a secure accumulator. The reduction will proceed in the following (straightforward) manner: On input $(\mathsf{f}, \mathsf{U}_\mathsf{f}, \mathsf{u})$, feed these values to the adversary. To respond to an ("ADD", x) query, simply update X and compute $\nu = \mathsf{f}(\mathsf{u}, \mathsf{X})$. To respond to a ("DELETE",x) query, compute $\nu = \mathsf{f}(\mathsf{u}, \mathsf{X} \setminus \{\mathsf{x}\})$, and then update X. The success of the adversary directly corresponds to the success of our reduction. □

Finally, in the application we have in mind we require that the accumulator allows for an efficient proof that a secret value given by some commitment is contained in a given accumulator value. That is, we require that the accumulator be *efficiently provable* with respect to some commitment scheme (*Commit*).

Zero-knowledge proof of member knowledge: There exists an efficient zero-knowledge proof of knowledge system where the common inputs are c (where $\mathsf{c} = \mathit{Commit}(\mathsf{x}, \mathsf{r})$ with a r being a randomly chosen string), the accumulating function f and $\nu \in \mathsf{U}_\mathsf{f}$, and the prover's inputs are (r, $\mathsf{x} \in \mathcal{X}_\mathsf{k}$, $\mathsf{u} \in \mathsf{U}_\mathsf{f}$) for proving knowledge of x, w, r such that $\mathsf{c} = \mathit{Commit}(\mathsf{x}, \mathsf{r})$ and $\nu = \mathsf{f}(w, \mathsf{x})$.

If by "efficient" we mean "polynomial-time," then any accumulator satisfies this property. However we consider a proof system efficient if it compares well with, for example, a proof of knowledge of a discrete logarithm.

3.2 Construction

The construction due to Barić and Pfitzmann [BP97] is the basis for our construction below. The differences from the cited construction are that (1) the domain of the accumulated values consists of prime numbers only; (2) we give a method for deleting values from the accumulator, i.e., we construct a *dynamic* accumulator; (3) we give efficient algorithms for deleting a user and updating a witness; and (4) we provide an efficient zero-knowledge proof of membership knowledge.

- $\mathcal{F}_k$ is the family of functions that correspond to exponentiating modulo safe-prime products drawn from the integers of length k. Choosing $f \in \mathcal{F}_k$ amounts to choosing a random modulus $n = pq$ of length k, where $p = 2p' + 1$, $q = 2q' + 1$, and p,p',q,q' are all prime. We will denote f corresponding to modulus n and domain $\mathcal{X}_{A,B}$ by $f_{n,A,B}$. We denote $f_{n,A,B}$ by f_n and by f when it does not cause confusion.
- $\mathcal{X}_{A,B}$ is the $\{e \in \texttt{primes} : e \neq p', q' \ \wedge \ A \leq e \leq B\}$, where A and B can be chosen with arbitrary polynomial dependence on the security parameter k, as long as $2 < A$ and $B < A^2$. $\mathcal{X}'_{A,B}$ is (any subset of) of the set of integer from $[2, A^2 - 1]$ such that $\mathcal{X}_{A,B} \subseteq \mathcal{X}'_{A,B}$.
- For $f = f_n$, the auxiliary information aux_f is the factorization of n.
- For $f = f_n$, $\mathcal{U}_f = \{u \in QR_n : u \neq 1\}$ and $\mathcal{U}'_f = \mathbb{Z}_n^*$.
- For $f = f_n$, $f(u, x) = u^x \bmod n$. Note that $f(f(u, x_1), x_2) = f(u, \{x_1, x_2\}) = u^{x_1 x_2} \bmod n$
- Update of the accumulator value. As mentioned earlier, adding a value $\tilde{x}$ to the accumulator value v can be done as $v' = f(v, \tilde{x}) = v^{\tilde{x}} \bmod n$. Deleting a value $\tilde{x}$ from the accumulator is as follows. $D((p, q), v, \tilde{x}) = v^{\tilde{x}^{-1} \bmod (p-1)(q-1)} \bmod n$.
- Update of witness: As mentioned, updating the witness u after $\tilde{x}$ has been added can be done by $u' = f(u, \tilde{x}) = u^{\tilde{x}}$. In case, $\tilde{x} \neq x \in \mathcal{X}_k$ has be deleted from the accumulator, the witness u can be updated as follows. By the extended GCD algorithm, one can compute the integers a,b such that $ax + b\tilde{x} = 1$ and then $u' = W(u, x, \tilde{x}, v, v') = u^b v'^a$. Let us verify that $f(u', x) = u'^x \bmod n = v'$:

$$\begin{aligned} (u^b v'^a)^x &= & (1)\\ ((u^b v'^a)^{x\tilde{x}})^{1/\tilde{x}} &= & (2)\\ ((u^x)^{b\tilde{x}} (v'^{\tilde{x}})^{ax})^{1/\tilde{x}} &= & (3)\\ (v^{b\tilde{x}} v^{ax})^{1/\tilde{x}} &= v^{1/\tilde{x}} = v' & (4) \end{aligned}$$

  Equation (2) is correct because $\tilde{x}$ is relatively prime to $\varphi(n)$.

We note that adding or deleting several values at once can be done simply by letting x' be the product of the added or deleted values. This also holds with respect to updating the witness. More precisely, let π_a be the product the x's to add to and π_d be the ones to delete from the accumulator value v. Then, the new accumulator value $v' := v^{\pi_a \pi_d^{-1} \bmod (p-1)(q-1)} \bmod n$. If u was the witness that x was contained in v and x was not removed from the accumulator, i.e., $x \nmid \pi_d$, then $u' u^{a\pi_a} v'^b \bmod n$ is a witness that x is contained in v', where a and b satisfy $ax + b\pi_d = 1$ and are computed using the extended GCD algorithm.

Theorem 3. *Under the strong RSA assumption, the above construction is a secure dynamic accumulator.*

Proof. Everything besides security is immediate. By Theorem 2, it is sufficient to show that the construction satisfies security as defined in Definition 1. Our proof is similar to the one given by Barić-Pfitzmann for their construction (the difference being that we do not require x' to be prime). The proof by Barić-Pfitzmann is actually the same as one given by Shamir [Sha83].

Suppose we are given an adversary $\mathcal{A}$ that, on input n and $u \in_R QR_n$, outputs m primes $x_1, \ldots, x_m \in \mathcal{X}_{A,B}$ and $u' \in \mathbb{Z}_n^*$, $x' \in \mathcal{X}'_{A,B}$ such that $(u')^{x'} = u^{\prod x_i}$. Let us use $\mathcal{A}$ to break the strong RSA assumption.

Suppose $n = pq$ that is a product of two safe primes, $p = 2p' + 1$ and $q = 2q' + 1$, is given. Suppose the value $u \in QR_n$ is given as well. To break the strong RSA assumption, we must output a value $e > 1$, y such that $y^e = u$.

We shall proceed as follows: Give (n, u) to the adversary. Suppose the adversary comes up with a forgery $(u', x', (x_1, \ldots, x_m))$. Let $x = \prod_{i=1}^m x_i$. Thus we have $u'^{x'} = u^x$.

Claim. *Let* $d = \gcd(x, x')$. *Then either* $d = 1$ *or* $d = x_j$ *for some* $1 \leq j \leq m$.

Proof of claim: Suppose $d|x$ and $d \neq 1$. Then, as $x_1, \ldots, x_m$ are primes, it follows that d is the product of a subset of primes. Suppose for some x_i and x_j we have $x_i x_j | d$. Then $x_i x_j | x'$. But this is a contradiction as $x_i x_j > x'$ must hold due to the definitions of $\mathcal{X}_{A,B}$ and $\mathcal{X}'_{A,B}$: Because $x' \in \mathcal{X}'_{A,B}$ we have $x' < A^2$. For any $x_i, x_j \in \mathcal{X}_{A,B}$, $x_i x_j \geq A^2 > x'$, as $x_1, x_2 \geq A$.

Back to the proof of the theorem: Suppose that $d \neq 1$ is not relatively prime to $\phi(n)$. Then, by the claim, for some j, $d = x_j$. Because $d = x_j \in \mathcal{X}_{A,B}$, $d > 2$ and d is prime. $\phi(n) = 4p'q'$, therefore $d = p'$ or $d = q'$. Then $2d + 1$ is a non-trivial divisor of n, so in this case we can factor n.

Suppose d is relatively prime to $\phi(n)$. Then, because $(u^{x/d})^d = ((u')^{x'/d})^d$, it follows that $u^{x/d} = (u')^{x'/d}$. Let $\tilde{x} = x/d$, and $\tilde{x}' = x'/d$. Because $\gcd(x, x') = d$, the equation $\gcd(\tilde{x}, \tilde{x}') = 1$ holds and thus one can compute a, b such that $a\tilde{x} + b\tilde{x}' = 1$ by extended GCD algorithm. Output $(y := \tilde{u}^a u^b, \tilde{x}')$. Note that $y^{\tilde{x}'} = (y^{\tilde{x}\tilde{x}'})^{1/\tilde{x}}((\tilde{u}^{\tilde{x}'})^{a\tilde{x}}(u^{\tilde{x}})^{b\tilde{x}'})^{1/\tilde{x}} = ((u^{\tilde{x}})^{a\tilde{x}+b\tilde{x}'})^{1/\tilde{x}} u$ and thus y and $\tilde{x}'$ are a solution to the instance (n, u) of the flexible RSA problem. $\square$

3.3 Efficient Proof That a Committed Value Was Accumulated

Here we show that the accumulator exhibited above is efficiently provable with respect to the Pedersen commitment scheme. Suppose that the parameters of the commitment scheme are a group $\mathfrak{G}_q$, and two generators $\mathfrak{g}$ and $\mathfrak{h}$ such that $\log_{\mathfrak{h}} \mathfrak{g}$ is unknown. Recall that to commit to a value $x \in \mathbb{Z}_q$, one picks a random $r \in_R \mathbb{Z}_q$ and outputs *Commit*$(x, r) := \mathfrak{g}^x \mathfrak{h}^r$. This information-theoretically hiding commitment scheme is binding under the discrete-logarithm assumption.

For the definitions of $\mathcal{X}_{A,B}$ and the choice of q, we require that $B2^{k'+k''+2} < A^2 - 1 < q/2$ holds, where k' and k'' are security parameters, i.e., k' is the bit length of challenges in the *PK* protocol below and k'' determines the statistical zero-knowledge property of the same protocol. We set $\mathcal{X}'_{A,B}$ the largest possible set, i.e., to $[2, A^2 - 1]$.

Finally, we require that two elements g and h of QR_n are available such that $\log_g h$ is not known to the prover, where n is the public key of the accumulator.

To prove that a given commitment $\mathfrak{C}_e$ and a given accumulator v contain the same (secret) value e, the following protocol is carried out. The common inputs to the protocol are the values $\mathfrak{C}_e$, $\mathfrak{g}$, $\mathfrak{h}$, n, g, h, and v. The prover's additional inputs are the values e, u, and r such that $u^e = v \bmod n$ and $\mathfrak{C}_e = \mathfrak{g}^e \mathfrak{h}^r$.

The prover will form a commitment C_u to u and prove that this commitment corresponds to the e-th root of the value v. This is carried out as follows:

1. The prover chooses $r_1, r_2, r_3 \in_R \mathbb{Z}_{\lfloor n/4 \rfloor}$, computes $C_e := g^e h^{r_1}$, $C_u := u h^{r_2}$, $C_r := g^{r_2} h^{r_3}$, and sends $\mathfrak{C}_e$, C_e, C_u, and C_r to the verifier.

2. The prover and verifier carry out the following proof of knowledge:

$$PK\{(\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \varphi, \psi, \eta, \sigma, \xi) :$$
$$\mathfrak{C}_e = \mathfrak{g}^{\alpha}\mathfrak{h}^{\varphi} \ \wedge \ \mathfrak{g} = (\frac{\mathfrak{C}_e}{\mathfrak{g}})^{\gamma}\mathfrak{h}^{\psi} \ \wedge \ \mathfrak{g} = (\mathfrak{g}\mathfrak{C}_e)^{\sigma}\mathfrak{h}^{\xi} \ \wedge$$
$$C_r = h^{\varepsilon} g^{\zeta} \ \wedge \ C_e = h^{\alpha} g^{\eta} \ \wedge \ v = C_u^{\alpha}(\frac{1}{h})^{\beta} \ \wedge \ 1 = C_r^{\alpha}(\frac{1}{h})^{\delta}(\frac{1}{g})^{\beta} \ \wedge$$
$$\alpha \in [-B2^{k'+k''+2}, B2^{k'+k''+2}] \ \} \ .$$

The details of this protocol can be found in Appendix A.

Theorem 4. *Under the strong RSA assumption the PK protocol in step 2 is a proof of knowledge of two integers* $e \in \mathcal{X}'_{A,B} = [2, A^2 - 1]$ *and* u *such that* $v \equiv u^e \pmod{n}$ *and* $\mathfrak{C}_e$ *is a commitment to* e.

Proof. Showing that the protocol is statistical zero-knowledge is standard. Also, it is easy to see that $\mathfrak{C}_e$, C_e, C_u, and C_r are statistically independent from u and e.

It remains to show that $\mathfrak{C}_e$ if the verifier accepts, then a value e and a witness w that e is in v can be extracted from the prover. Using standard rewinding techniques, the knowledge extractor can get answers $(s_\alpha, s_\beta, s_\gamma, s_\delta, s_\varepsilon, \ s_\zeta, s_\eta, s_\varphi, s_\psi)$ and $(s'_\alpha, s'_\beta, s'_\gamma, s'_\delta, s'_\varepsilon, s'_\eta, s'_\zeta, s'_\varphi, s'_\psi)$ for the two different challenges c and c'. Let $\Delta\alpha = s_\alpha - s'_\alpha$, $\Delta\beta = s_\beta - s'_\beta$, $\Delta\gamma = s_\gamma - s'_\gamma$, $\Delta\delta = s_\delta - s'_\delta$, $\Delta\varepsilon = s_\varepsilon - s'_\varepsilon$, $\Delta\zeta = s_\zeta - s'_\zeta$, $\Delta\eta = s_\eta - s'_\eta$, $\Delta\varphi = s_\varphi - s'_\varphi \bmod \mathfrak{q}$, $\Delta\psi = s_\psi - s'_\psi$, $\Delta\sigma = s_\sigma - s'_\sigma$, $\Delta\xi = s_\xi - s'_\xi$, and $\Delta c = c' - c$. Then we have

$$\mathfrak{C}_e^{\Delta c} = \mathfrak{g}^{\Delta\alpha}\mathfrak{h}^{\Delta\varphi} \ , \qquad \mathfrak{g}^{\Delta c} = (\frac{\mathfrak{C}_e}{\mathfrak{g}})^{\Delta\gamma}\mathfrak{h}^{\Delta\psi} \ , \qquad \mathfrak{g}^{\Delta c} = (\mathfrak{g}\mathfrak{C}_e)^{\Delta\sigma}\mathfrak{h}^{\Delta\xi} \tag{5}$$
$$C_r^{\Delta c} = h^{\Delta\varepsilon} g^{\Delta\zeta} \ , \qquad C_e^{\Delta c} = h^{\Delta\alpha} g^{\Delta\eta} \ , \tag{6}$$
$$v^{\Delta c} = C_u^{\Delta\alpha}(\frac{1}{h})^{\Delta\beta} \ , \qquad 1 = C_r^{\Delta\alpha}(\frac{1}{h})^{\Delta\delta}(\frac{1}{g})^{\Delta\beta} \ . \tag{7}$$

We first show that $\mathfrak{C}_e$ commits to an integer different from 1 and consider the first two equations (5). Let $\widetilde{\alpha} := \Delta\alpha\Delta c^{-1} \bmod \mathfrak{q}$, $\widetilde{\gamma} := \Delta\gamma\Delta c^{-1} \bmod \mathfrak{q}$, $\widetilde{\varphi} := \Delta\varphi\Delta c^{-1} \bmod \mathfrak{q}$, and $\widetilde{\psi} := \Delta\psi\Delta c^{-1} \bmod \mathfrak{q}$. Then we have

$$\mathfrak{C}_e = \mathfrak{g}^{\widetilde{\alpha}}\mathfrak{h}^{\widetilde{\varphi}} \quad \text{and} \quad \mathfrak{g} = (\frac{\mathfrak{C}_e}{\mathfrak{g}})^{\widetilde{\gamma}}\mathfrak{h}^{\widetilde{\psi}} = \mathfrak{g}^{(\widetilde{\alpha}-1)\widetilde{\gamma}}\mathfrak{h}^{\widetilde{\varphi}\widetilde{\gamma}}\mathfrak{h}^{\widetilde{\psi}} \ .$$

Under the hardness of computing discrete logarithms, $1 \equiv (\widetilde{\alpha} - 1)\widetilde{\gamma} \pmod{\mathfrak{q}}$ must hold and therefore $\widetilde{\alpha} \not\equiv 1 \pmod{\mathfrak{q}}$ as otherwise $\widetilde{\gamma}$ would not exists. Similarly, from the first and third equation of (5) one can conclude that $\widetilde{\alpha} \not\equiv -1 \pmod{\mathfrak{q}}$.

We next show that $\widetilde{\alpha}$ is accumulated in v. From the next two equations (6) one can derive that Δc divides $\Delta\alpha$, $\Delta\eta$, $\Delta\varepsilon$, and $\Delta\zeta$ provided the strong RSA assumption. (While we do not investigate this claim here, one can show that if Δc does not divide $\Delta\alpha$, $\Delta\eta$, $\Delta\varepsilon$, and $\Delta\zeta$, then from the Equations (6) one can compute a non-trivial root of g with probability at least $1/2$. This, however, is not feasible under the strong RSA assumption. We refer to, e.g., [DF01] for the details of such a reduction.) Let $\hat{\alpha} = \Delta\alpha/\Delta c$, $\hat{\eta} = \Delta\eta/\Delta c$, $\hat{\varepsilon} = \Delta\varepsilon/\Delta c$ and $\hat{\zeta} = \Delta\zeta/\Delta c$. Because $|c|, |c'| < p', q'$, we get $C_r = a h^{\hat{\varepsilon}} g^{\hat{\zeta}}$ for some a such that $a^2 = 1$. Moreover, the value

a must be either 1 or -1 as otherwise $1 < \gcd(a-1, n) < n$ and we could factor n. Plugging C_r into the second equation of (7) we get

$$1 = a^{\Delta\alpha} h^{\Delta\alpha\hat{\varepsilon}} g^{\Delta\alpha\hat{\zeta}} (\frac{1}{h})^{\Delta\delta} (\frac{1}{g})^{\Delta\beta} \ ,$$

where $a^{\Delta\alpha}$ must be 1 as 1, g, and h are in QR_n and $a^2 = 1$ otherwise. Under the hardness of computing discrete logarithms we can conclude that $\Delta\alpha\hat{\zeta} \equiv \hat{\beta} \pmod{\operatorname{ord}(g)}$ and hence we get

$$v^{\Delta c} = (\frac{C_u}{h^{\hat{\zeta}}})^{\Delta\alpha} \quad \text{and} \quad v = b(\frac{C_u}{h^{\hat{\zeta}}})^{\hat{\alpha}}$$

with some b such that $b^2 = 1$. Again $b = \pm 1$ as otherwise $1 < \gcd(b \pm 1, n) < n$ and we could factor n. Let $s = -1$ if $\hat{\alpha} < 0$ and $s = 1$. Thus we have $v = u^{|\hat{\alpha}|}$,

$$u = \begin{cases} (b\frac{C_u}{h^{\hat{\zeta}}})^s & \text{if } \hat{\alpha} \text{ is odd} \\ (\frac{C_u}{h^{\hat{\zeta}}})^s & \text{if } \hat{\alpha} \text{ is even.} \end{cases}$$

The latter holds because $v \in QR_n$ and $-1 \notin QR_n$ and therefore $b = -1$ is not possible. Also note that $\hat{\alpha} \neq 0$ as $v \neq 1$. Because $s_\alpha, s'_\alpha \in [-B2^{k'+k''+1}, -B2^{k'+k''+1}]$ we have $\Delta\alpha, \hat{\alpha} \in [-B2^{k'+k''+2}, -B2^{k'+k''+2}]$. Because $B2^{k'+k''+2} < q/2$ it follows that $\hat{\alpha} = (\Delta\alpha\hat{c} \operatorname{rem} q)(\widetilde{\alpha} \operatorname{rem} q)$, and hence that the absolute value committed to by $\mathfrak{C}_e$ is indeed accumulated in v. As $B2^{k'+k''+2} < A^2 - 1$, $\hat{\alpha} \neq \pm 1 \mod q$ and $\hat{\alpha} \neq 0$ we can conclude that $|\hat{\alpha}| \in \mathcal{X}'_{A,B}$. Therefore, due to Theorem 3, we can conclude that $|\hat{\alpha}|$ must be contained in the accumulator value v.

$\square$

4 Application to ID Escrow, Group Signatures and Credential Systems

In this section we describe how dynamic accumulators can be used to obtain membership revocation for identity escrow and group signature schemes and credential systems. In particular, we provide an efficient identity escrow scheme with membership revocation. We first informally discuss the properties of identity escrow schemes with membership revocation. The translation to group signatures scheme and credential systems is straightforward.

An identity escrow scheme with membership revocation consists of the following procedures:

Setup: An algorithm for the group manager to generate the system parameters, the group's public key, and her secret key.

Join: A protocol between a group member and the group manager. Their common input is the group's public key. Their common output is the user's membership public key and membership certificate. The user's output is the membership secret key. In addition, the group manager gets some information to be made available in a public archive as well as an updated version of the group's public key.

Prove membership: A protocol between a group member and a verifier (whose sole input consist of the group's public key) that allows the former to convince the latter of his membership in the group.

Anonymity revocation: A procedure that allows the group manager on input her secret key and a verifier's transcript of the membership-proof protocol outputs the membership public key of the user with whom the verifier ran the protocol in question.

Membership revocation: A procedure for the group manager to remove a member from the group. This procedure results in an updated group's public key as well as some information to be made available in a public archive.

Membership update: A procedure for the group members to update their membership certificates using the information available in the public archives and the current public key of the group.

The scheme must provide the following properties.

Correctness: The scheme functions properly if all participants are honest.

Unforgeability and traceability: Upon revocation of its anonymity, each transcript of a successful membership-proof reveals the identity of a user who was a member of the group at the time the protocol in question took place.

Exculpability: It is infeasible (even to the group manager) to make it appear that an honest user participated in a membership-proof if he did not.

Anonymity and unlinkability: Linking a transcript of membership-proof protocol (run with a possible dishonest verifier) to a user is computationally infeasible to everyone but the group manager and so is determining whether two transcripts stem from the same or from different users. We stress that in case a user's membership is revoked, anonymity and unlinkability is retained for transcripts stemming from the interaction with that user prior to his membership revocation.

All of the above properties must hold even in the presence of an adversary that is allowed to run all the protocols and procedures adaptively with the honest parties. This can be made formal in an ideal-world/real-world model (c.f. [Can95, Can00, PW00]) similarly as is done by Camenisch and Lysyanskaya [CL01b] for identity escrow schemes.

4.1 Overview of Efficient Group Signatures and Credential Systems

Recall the ACJT [ACJT00] identity escrow scheme. (Recall that the ACJT group signature scheme is obtained from the ACJT identity escrow by applying the Fiat-Shamir heuristic to the protocol for proving membership.) The group manager has a public key PK, consisting of a number $\mathfrak{n}$, which is a product of two safe primes, the values $\mathfrak{a}$, $\mathfrak{b}$, $\mathfrak{g}$, $\mathfrak{h}$, and $\mathfrak{y}$ which are quadratic residues modulo $\mathfrak{n}$, and two intervals Γ and Δ. The value $z = \log_{\mathfrak{g}} \mathfrak{y}$ is a secret key of the group manager used for revocation. A user U_i's membership certificate consists of a user's secret x_i selected jointly by the user and the group manager (it is selected in a secure manner that ensures that the group manager obtains no information about this value) from an appropriate integer range, i.e., Δ, and the values $\mathfrak{v}_i$ and e_i, where e_i is a prime number selected from another appropriate range, i.e., Γ, and $\mathfrak{v}_i^{e_i} = \mathfrak{a}^{x_i}\mathfrak{b} \bmod \mathfrak{n}$. The value $\mathfrak{a}^{x_i}$ is user U_i's public key. When U_i proves membership in a group, he effectively proves knowledge of a membership certificate $(x, \mathfrak{v}, e)$. This proof is as follows. The group member chooses $r_1', r_2' \in_R \in_R \mathbb{Z}_{\lfloor \mathfrak{n}/4 \rfloor}$ and computes $\mathfrak{T}_1 := \mathfrak{v}\mathfrak{y}^{r_1'}$, $\mathfrak{T}_2 := \mathfrak{g}^{r_1'}$, and $\mathfrak{T}_3 := \mathfrak{g}^{e}\mathfrak{h}^{r_2'}$. The group member sends $\mathfrak{T}_1$, $\mathfrak{T}_2$, and $\mathfrak{T}_3$ to the verifier and carries out with the verifier the protocol denoted

$$PK\Big\{(\alpha, \beta, \gamma, \delta, \varepsilon) : \ \mathfrak{b} = \mathfrak{T}_1^{\alpha}\Big(\frac{1}{\mathfrak{a}}\Big)^{\beta}\Big(\frac{1}{\mathfrak{y}}\Big)^{\gamma} \ \wedge \ 1 = \mathfrak{T}_2^{\alpha}\Big(\frac{1}{\mathfrak{g}}\Big)^{\gamma} \ \wedge \ \mathfrak{T}_2 = \mathfrak{g}^{\delta} \ \wedge \ \mathfrak{T}_3 = \mathfrak{g}^{\alpha}\mathfrak{h}^{\varepsilon} \ \wedge \\ \alpha \in \Gamma \ \wedge \ \beta \in \Delta\Big\} \ .$$

As with all group signature and identity escrow schemes, only the group manager can assert a signature/protocol transcript to a group member, i.e., knowing z, the group manager can compute the value $\hat{\mathfrak{v}} = \mathfrak{T}_1/\mathfrak{T}_2{}^z$ that identifies the user.

The Camenisch and Lysyanskaya [CL01a] credential system has a similar construction. An organization's public key consists of a number $\mathfrak{n}$, which is a product of two safe primes, and the values $\mathfrak{a}$, $\mathfrak{b}$, $\mathfrak{c}$, $\mathfrak{g}$ and $\mathfrak{h}$ which are all quadratic residues modulo $\mathfrak{n}$. A user U_i's secret key x_i, selected from an appropriate integer range, is incorporated into all of U_i's credentials. A credential tuple for user U_i consists of his secret key x_i, a secret value s_i selected jointly by the U_i and the organization (via a secure computation which ensures secrecy for the user) from an appropriate integer range, and the values $\mathfrak{v}_i$ and e_i such that e_i is a prime number selected by the organization from an appropriate integer interval, and $\mathfrak{v}_i$ is such that $\mathfrak{v}_i^{e_i} = \mathfrak{a}^x \mathfrak{b}^s \mathfrak{c} \bmod N$. Proving possession of a credential is effectively a proof of knowledge of a credential tuple.

Variations of these schemes incorporate such features as anonymity revocation, non-transferability, one-show credentials, expiration dates, and appointed verifiers. For all these variations, an integral part of a group membership certificate and of a credential, is the prime number e_i. Using one-way accumulators, we can accumulate e_i's into a single public value u. Proof of group membership will now have to include proof of knowledge of a witness to the fact that e_i was accumulated into u.

In the sequel, we will talk about augmenting the ACJT identity escrow scheme with the membership revocation property; however, all our results and discussion applies immediately to the credential scheme and group signature discussed above.

4.2 Incorporating Revocation into the ACJT Identity Escrow Scheme

To make certificate revocation possible, the additions outlined below have to be made to the usual operations the ACJT identity escrow scheme.

Modifications to the group manager's operations are as follows:

Setup: In addition to setting up the identity escrow scheme, the group manager creates the public modulus n for the accumulator, chooses a random $u, g, h \in QR_n$ and publishes (n, u, g, h). She sets up (empty for now) public archives E_{add} for storing values that correspond to added users and E_{delete} for storing values that correspond to deleted users. Set $\mathcal{X}'_{A,B} = \Gamma$ and $\mathcal{X}_{A,B}$ to the interval from which the group manager chooses e in the ACJT scheme ($\mathcal{X}_{A,B} \subseteq \mathcal{X}'_{A,B} \subseteq [2, A^2 - 1]$ will be satisfied).

Join: Issue the user's membership certificate, as in the identity escrow scheme. Add the current u to the user's membership certificate. (Denote it by u_i.) Let e_i be the prime number used in this certificate. Update u in the public key: $u := f_n(u, e_i)$. Update E_{add}: store e_i there.

Revoke membership: Retrieve e_i which is the prime number corresponding to the user's membership certificate. Update u in the public key: $u := D(\varphi(n), u, e_i)$. Update E_{delete}: store e_i there.

We stress that the archives are E_{add} and E_{delete} are *not* part of the group's public key, i.e., the verifier is not required to read them for any verification purposes. Also, note that is it not necessary to restrict read access to these archives only to group members.

A user U_i must augment the ACJT protocol as follows:

Join: Store the value u_i along with the rest of the membership certificate. Verify that $f_n(u_i, e_i) = u_i^{e_i} = u$.

Update membership: An entry in the archive is called "new" if it was entered after the last time U_i performed an update.

1. Let y denote the old value of u.
2. For all new $e_j \in E_{\text{add}}$, $u_i := f(u_i, \prod e_j) = u_i^{\prod e_j}$ and $y := y^{\prod e_j}$
3. For all new $e_j \in E_{\text{delete}}$, $u_i := W(u_i, e_i, \prod e_j, y, u)$.

(Note that as a result $u = f(u_i, e_i)$.)

Prove membership: Proving membership is augmented with the step of proving that a committed value is part of the accumulated value v (contained in the current public key). That is, in addition to $\mathfrak{T}_1$, $\mathfrak{T}_2$, and $\mathfrak{T}_3$ the group member computes the values $C_e := g^e h^{r_1}$, $C_u := u h^{r_2}$, and $C_r := g^{r_2} h^{r_3}$ and sends them to verifier, with random choices $r_1, r_2, r_3 \in_R \mathbb{Z}_{\lfloor n/4 \rfloor}$. Then the verifier and the group member engage in the protocol denoted

$$\begin{aligned}
&PK\{(\alpha, \beta, \gamma, \delta, \varepsilon, \xi, \zeta, \varphi, \psi, \eta) : \\
&\qquad \mathfrak{w} \equiv \mathfrak{T}_1^{\alpha}(\frac{1}{\mathfrak{a}})^{\beta}(\frac{1}{\mathfrak{y}})^{\gamma} \ \wedge\ 1 \equiv \mathfrak{T}_2^{\alpha}(\frac{1}{\mathfrak{g}})^{\gamma} \ \wedge\ \mathfrak{T}_2 \equiv \mathfrak{g}^{\delta} \ \wedge\ \mathfrak{T}_3 \equiv \mathfrak{g}^{\alpha}\mathfrak{h}^{\varepsilon} \ \wedge \\
&\qquad C_r = h^{\xi} g^{\zeta} \ \wedge\ C_e = h^{\alpha} g^{\eta} \ \wedge\ v = C_u^{\alpha}(\frac{1}{h})^{\varphi} \ \wedge\ 1 = C_r^{\alpha}(\frac{1}{h})^{\psi}(\frac{1}{g})^{\varphi} \ \wedge \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \alpha \in \Gamma \ \wedge\ \beta \in \Delta\} \ .
\end{aligned}$$

This protocol is already an optimized union of the *PK* protocol given in the previous section and the ACJT *PK* protocol for proving group membership. That is, different from the previous section, we do not require the group $\mathfrak{G}_q$ for the commitment scheme because here the value $\mathfrak{T}_3$ acts as commitment to the value whose membership in the accumulator is claimed. Furthermore, as $-1, 0, 1 \notin \Gamma$, we need not show that $\alpha \neq -1, 0, 1$.

The complexity of this augmented proof is about twice that of the original one. The definition of Γ is compatible with the accumulator and the proof that a committed value is contained in the accumulator as presented in the previous section. Also, Γ excludes 1 and hence it is not required to explicitly prove that the committed value is not 1.

Remark. Updates after a users joined the group can be avoided if the group managers chooses all the e_i at setup-time and already accumulates them, i.e., $u := f_n(u, \prod e_i)$. Note that the group manager can always compute the witness for e_i as u^{1/e_i}. If this is done, only deletion of member requires updates by the group manager and the group members (or if the group manager runs out of e_i's).

Lemma 5. *Under the strong RSA assumption the above is a secure identity escrow scheme with membership revocation.*

Proof (sketch). It is not hard to show the security of this lemma in a formal model given the security proofs of the ACJT scheme and the proof of Theorem 4. Let us provide an informal argument here.

First of all, note that all the properties of the original ACJT scheme are retained as the amount of information revealed by C_e, C_u, and C_r about the group member's certificate is negligible (i.e., C_e, C_u, and C_r are statistically hiding commitments and the *PK*-protocol is statistical zero-knowledge). It remains to argue that excluded group members can no longer prove group membership even if they collude in an adaptive attack against the group manager. Similarly as in the proof of Theorem 4, one can show that the above of a protocol is a proof of knowledge of a quadruple $(\hat{x}, \hat{\mathfrak{v}}, \hat{e}, \hat{u})$ such that $\mathfrak{a}^{\hat{x}}\mathfrak{b} = \hat{\mathfrak{v}}^{\hat{e}}$ and $\hat{u}^{\hat{e}} = u$ hold, i.e., such that $(\hat{x}, \hat{\mathfrak{v}}, \hat{e})$ is valid group membership certificate and $\hat{e}$ is contained in the accumulator value u. In [ACJT00], Ateniese et al. show that under the strong RSA assumption an adaptive adversary controlling all users cannot find a triple $(\tilde{x}, \tilde{\mathfrak{v}}, \tilde{e})$ that is different from the ones legitimately obtained through the *join* protocol. On other words, the values $\mathfrak{a}^{x_i}$ and e_i are tightly linked. Therefore, the user with public key $\mathfrak{a}^{x_i}$ is no longer able to prove membership of the group once an e_i is removed from the accumulator value as the accumulator is secure against an adaptive adversary (Theorem 2). We note that all these arguments hold in spite of the fact that all members' (current and past one) e_i's are public. It follows that anonymity and unlinkability is retained for actions past members made prior to their exclusion from the group. □

References

[ACJT00] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *LNCS*, pages 255–270. Springer Verlag, 2000.

[AT01] Giuseppe Ateniese and Gene Tsudik. Quasi-efficient revocation of group signatures. `http://eprint.iacr.org/2001/101`, 2001.

[BdM94] Josh Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, volume 765 of *LNCS*, pages 274–285. Springer-Verlag, 1994.

[BP97] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *LNCS*, pages 480–494. Springer Verlag, 1997.

[BS01] Emmanuell Bresson and Jacques Stern. Group signatures with efficient revocation. In Kwangjo Kim, editor, *Proceedings of 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC2001*, volume 1992 of *LNCS*, pages 190–206. Springer, 2001.

[Cam97] Jan Camenisch. Efficient and generalized group signatures. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *LNCS*, pages 465–479. Springer Verlag, 1997.

[Can95] Ran Canetti. *Studies in Secure Multiparty Computation and Applications.* PhD thesis, Weizmann Institute of Science, Rehovot 76100, Israel, June 1995.

[Can00] Ran Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[CL01a] Jan Camenisch and Anna Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer Verlag, 2001.

[CL01b] Jan Camenisch and Anna Lysyanskaya. An identity escrow scheme with appointed verifiers. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *LNCS*, pages 388–407. Springer Verlag, 2001.

[CM98] Jan Camenisch and Markus Michels. A group signature scheme with improved efficiency. In Kazuo Ohta and Dinqyi Pei, editors, *Advances in Cryptology — ASIACRYPT '98*, volume 1514 of *LNCS*, pages 160–174. Springer Verlag, 1998.

[CM99] Jan Camenisch and Markus Michels. Separability and efficiency for generic group signature schemes. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *LNCS*, pages 413–430. Springer Verlag, 1999.

[CP95] Lidong Chen and Torben Pryds Pedersen. New group signature schemes. In Alfredo De Santis, editor, *Advances in Cryptology — EUROCRYPT '94*, volume 950 of *LNCS*, pages 171–181. Springer-Verlag, 1995.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In Burt Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1296 of *LNCS*, pages 410–424. Springer Verlag, 1997.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1642 of *LNCS*, pages 13–25, Berlin, 1998. Springer Verlag.

[DF01] Ivan Damgård and Eiichiro Fujisaki. An integer commitment scheme based on groups with hidden order. `http://eprint.iacr.org/2001`, 2001.

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burt Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *LNCS*, pages 16–30. Springer Verlag, 1997.

[GHR99] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *LNCS*, pages 123–139. Springer Verlag, 1999.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86*, volume 263 of *LNCS*, pages 171–185. Springer-Verlag, 1987.

[KLL01] Hyun-Jeong Kim, Jong In Lim, and Dong Hoon Lee. Efficient and secure member deletion in group signature schemes. In D. Won, editor, *ICISC 2000*, number 2015 in LNCS, pages 150–161. Springer Verlag, 2001.

[KP98] Joe Kilian and Erez Petrank. Identity escrow. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1642 of *LNCS*, pages 169–185, Berlin, 1998. Springer Verlag.

[PW00] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *Proc. 7th ACM Conference on Computer and Communications Security*, pages 245–254. ACM press, nov 2000.

[Sha83] Adi Shamir. On the generation of cryptographically strong pseudorandom sequences. In *ACM Transaction on Computer Systems*, volume 1, pages 38–44, 1983.

[Son01] Dawn Xiaodong Song. Practical forward secure group signature schemes. In *Proc. 8th ACM Conference on Computer and Communications Security*, pages 225–234. ACM press, nov 2001.

A Protocol to Prove that a Committed Value is Accumulated

This section provides the details of the protocol denoted

$$\begin{aligned}PK\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\varphi,\psi,\eta):\ &\mathfrak{C}_e=\mathfrak{g}^{\alpha}\mathfrak{h}^{\varphi}\ \wedge\ \mathfrak{g}=(\frac{\mathfrak{C}_e}{\mathfrak{g}})^{\gamma}\mathfrak{h}^{\psi}\ \wedge\ C_r=h^{\varepsilon}g^{\zeta}\ \wedge\\ C_e=h^{\alpha}g^{\eta}\ \wedge\ v=C_u^{\alpha}(\frac{1}{h})^{\beta}\ \wedge\ 1=C_r^{\alpha}(\frac{1}{h})^{\delta}(\frac{1}{g})^{\beta}\ &\wedge\ \alpha\in[-B2^{k'+k''+2},B2^{k'+k''+2}]\ \}\ .\end{aligned}$$

that can be used (as described in §3.3) to prove that value committed to in $\mathfrak{C}_e$ is accumulated in v. The values C_u, C_e and C_r are auxiliary commitments (c.f. §3.3).

1. The prover chooses

$$\begin{gathered}r_\alpha\in_R(-B2^{k'+k''},\dots,B2^{k'+k''})\ ,\\ r_\gamma,r_\varphi,r_\psi,r_\sigma,r_\xi\in_R\mathbb{Z}_q\ ,\\ r_\varepsilon,r_\eta,r_\zeta\in_R(-\lfloor n/4\rfloor 2^{k'+k''},\dots,\lfloor n/4\rfloor 2^{k'+k''})\ ,\ \text{and}\\ r_\beta,r_\delta\in_R(-\lfloor n/4\rfloor q2^{k'+k''},\dots,\lfloor n/4\rfloor q2^{k'+k''})\ ,\end{gathered}$$

   computes

$$\begin{aligned}&\mathfrak{t}_1:=\mathfrak{g}^{r_\alpha}\mathfrak{h}^{r_\varphi}\ , && \mathfrak{t}_2:=(\frac{\mathfrak{C}_e}{\mathfrak{g}})^{r_\gamma}\mathfrak{h}^{r_\psi}\ , && \mathfrak{t}_3:=(\mathfrak{g}\mathfrak{C}_e)^{r_\sigma}\mathfrak{h}^{r_\xi}\ ,\\ &t_1:=h^{r_\varepsilon}g^{r_\zeta}\ , && t_2:=h^{r_\alpha}g^{r_\eta}\ , && t_3:=C_u^{r_\alpha}(\frac{1}{h})^{r_\beta}\ ,\ \text{and}\\ &t_4:=C_r^{r_\alpha}(\frac{1}{h})^{r_\delta}(\frac{1}{g})^{r_\beta}\end{aligned}$$

   and sends $\mathfrak{t}_1$, $\mathfrak{t}_2$, $\mathfrak{t}_3$, t_1, t_2, t_3, and t_4 to the verifier.

2. The verifier chooses $c\in_R\{0,1\}^k$ and sends it to the prover.

3. The prover computes

$$\begin{aligned}&s_\alpha:=r_\alpha-ce\ , && s_\eta:=r_\beta-cr_1\ , && s_\varphi:=r_\varphi-cr \bmod q\ ,\\ &s_\beta:=r_\beta-cr_2e\ , && s_\varepsilon:=r_\varepsilon-cr_2\ , && s_\gamma:=r_\gamma-c(e-1)^{-1} \bmod q\ ,\\ &s_\zeta:=r_\zeta-cr_3\ , && s_\delta:=r_\delta-cr_3e\ , && s_\psi:=r_\psi-cr(e-1)^{-1} \bmod q\ ,\\ &s_\sigma:=r_\sigma-c(e+1)^{-1} \bmod q\ ,\ \text{and} && s_\xi:=r_\xi-cr(e+1)^{-1} \bmod q\end{aligned}$$

   and sends them to the verifier.

4. The verifier accepts if the following equations hold:

$$t_1 \overset{?}{=} \mathfrak{C}_e^c \mathfrak{g}^{s_\alpha} \mathfrak{h}^{s_\varphi}, \qquad t_2 \overset{?}{=} \mathfrak{g}^c \left(\frac{\mathfrak{C}_e}{\mathfrak{g}}\right)^{s_\gamma} \mathfrak{h}^{s_\psi}\ , \qquad t_3 \overset{?}{=} \mathfrak{g}^c (\mathfrak{g}\mathfrak{C}_e)^{s_\sigma} \mathfrak{h}^{s_\xi}\ ,$$

$$t_1 \overset{?}{=} C_r^c h^{s_\varepsilon} g^{s_\zeta}\ , \qquad t_2 \overset{?}{=} C_e^c h^{s_\alpha} g^{s_\eta}\ , \qquad t_3 \overset{?}{=} v^c C_u^{s_\alpha} \left(\frac{1}{h}\right)^{s_\beta}\ ,$$

$$t_4 \overset{?}{=} C_r^{s_\alpha} \left(\frac{1}{h}\right)^{s_\delta} \left(\frac{1}{g}\right)^{s_\beta}\ , \text{ and } \quad s_\alpha \overset{?}{\in} [-B2^{k'+k''+1}, B2^{k'+k''+1}]\ .$$