Universal Accumulators with Efficient Nonmembership Proofs

Jiangtao Li[1], Ninghui Li[2], and Rui Xue[3]

[1] Intel Corporation
jiangtao.li@intel.com
[2] Purdue University
ninghui@cs.purdue.edu
[3] University of Illinois at Urbana-Champaign
rxue@uiuc.edu

Abstract. Based on the notion of accumulators, we propose a new cryptographic scheme called universal accumulators. This scheme enables one to commit to a set of values using a short accumulator and to efficiently compute a membership witness of any value that has been accumulated. Unlike traditional accumulators, this scheme also enables one to efficiently compute a nonmembership witness of any value that has not been accumulated. We give a construction for universal accumulators and prove its security based on the strong RSA assumption. We further present a construction for dynamic universal accumulators; this construction allows one to dynamically add and delete inputs with constant computational cost. Our construction directly builds upon Camenisch and Lysyanskaya's dynamic accumulator scheme. Universal accumulators can be seen as an extension to dynamic accumulators with support of nonmembership witness. We also give an efficient zero-knowledge proof protocol for proving that a committed value is not in the accumulator. Our dynamic universal accumulator construction enables efficient membership revocation in an anonymous fashion.

1 Introduction

Accumulators were first introduced by Benaloh and de Mare [4] as a method to condense a set of values into one short accumulator, such that there is a short witness for each value that has been accumulated. In the mean time, it is infeasible to find a witness for a value that has not been accumulated. Barić and Pfitzmann [3] proposed a construction of a collision-resistant accumulator under the strong RSA assumption. Camenisch and Lysyanskaya [10] further proposed a dynamic accumulator in which elements can be efficiently added into or removed from the accumulator. Accumulators have been used in many applications [3, 4, 10, 20], including membership testing, time stamping, authenticated directory, and certificate revocation.

None of the existing accumulator schemes provide nonmembership witnesses for values that have not been accumulated. This feature of nonmembership witnesses is highly desirable in many applications. The following are two examples where a nonmembership witness is important.

1. Suppose a credit report agency compiles a list of users who have gone into bankruptcy within the last three years, and it also publishes the accumulator for this list. When Alice applies an auto loan from a bank, the bank wants Alice to prove that she is not in the bankruptcy list. In a similar application, suppose a Certificate Authority (CA) revokes a number of certificates before their expiration dates. A certificate user may need to efficiently prove that her certificate has not been revoked when using her certificate.
2. Suppose the Center for Disease Control and Prevention maintains a list of patients who have a certain infectious disease (e.g., Measles or Cholera). In some applications, a patient needs to prove that she has the disease in order to purchase discounted medicines from the local pharmacy stores. Whereas in other applications, one needs to prove that she does not have the disease.

In this paper, we propose the notion of *universal accumulators*, which enables a trusted group manager to condense a list of values into a short accumulator. For each value in the list, there is a short membership witness; and for each value not in the list, there exists a short nonmembership witness. It is computationally infeasible to find a membership witness for a value that was not accumulated or to find a nonmembership witness for a value that was accumulated. The notion of universal comes from the fact that each possible value in the input domain has a witness (either a membership witness or a nonmembership witness). Using universal accumulators, one can easily solve the problems in the aforementioned applications.

We further propose the notion of *dynamic universal accumulators*, which allow one to dynamically add and delete inputs, such that the the cost of an addition or a deletion is independent of the size of the accumulated set. We construct an efficient dynamic universal accumulator under the strong RSA assumption. Dynamic universal accumulators enable efficient membership revocation: A group manager issues a credential for each group member. The group manager also maintains a credential revocation list using a dynamic universal accumulator. To revoke a member, the group manager simply inserts the serial number of the revoked credential into the revoked list. To prove membership, a valid group member first shows her group credential, then shows that the credential's serial number is not in the revocation list by presenting the nonmembership witness.

We also develop an efficient zero-knowledge proof protocol such that, if a value is stored in a cryptographic commitment, then one can prove that the value is not accumulated in a zero-knowledge fashion. This enables membership revocation in an anonymous setting. To prove membership anonymously, the group member first proves that she has a valid group credential, then proves that the credential's serial number is not in the revocation list.

Note that many applications that require nonmembership proofs (such as ones in Application 1) can be solved using membership-proof techniques. Take the certificates revocation as an example, instead of proving the nonmembership of a revocation list, one can prove the membership of a legitimate user list to show that her certificate has not been revoked. This idea was used by, e.g., Camenisch and Lysyanskaya [10]. However, even though proving membership is efficient in [10], the maintenance overhead of the witness could be very expensive if the user list is huge and frequently-changing. For example, consider a certificate system that has thousands or millions of users. Suppose the list of valid users increases every hour (i.e., each hour there are several new users added into the list). And suppose the list of revoked users is small in size and relatively static (e.g., changes every month). Using the scheme in [10], a legitimate user may have to update her witness every hour, whereas using our scheme, she only need to update her nonmembership witness once a month.

1.1 Our contribution

The contributions of this paper are as follows.

- We introduce the notion of universal accumulators, which support short witnesses for both membership and nonmembership.
- We construct an efficient dynamic universal accumulator based on the strong RSA assumption. The update of witness in our scheme can be efficiently performed without the help of the trusted group manager. Proofs of membership or nonmembership can be achieved with a constant number of modular exponentiations.
- We give an efficient zero-knowledge proof protocol to prove that a committed value was not accumulated. This enables efficient membership revocation in anonymous credential systems, group signature schemes, and direct anonymous attestation schemes. Universal accumulators may be of interest in other applications as well.

1.2 Organization of this paper

The rest of this paper is organized as follows. We first give notations and security assumptions in section 2. In section 3, we give a formal definition of universal accumulators and present our construction. In section 4, we present the notion of dynamic universal accumulators and describe the corresponding construction. In section 5, we present a zero-knowledge proof protocol to prove that a committed value has not been accumulated in an accumulator. In section 6, we discuss several applications of dynamic universal accumulators to membership revocations in the anonymous setting, we also compare our solution with other existing membership revocation techniques.We conclude our paper in section 7.

2 Notations and Assumptions

2.1 Notations

We use $\phi(\cdot)$ to denote the Euler totient function. Let $n = pq$ be a RSA modulus, we use $\mathbb{Z}_n^*$ to denote the set of all positive integers that are less than n and relative prime to n. We use QR_n to denote the set of quadratic residues modulo n.

A negligible function, denoted by $\mathrm{neg}(\cdot)$, represents a positive function that vanishes faster than the inverse of any fixed positive polynomial. That is, for every polynomial $p(\cdot)$ and for every large enough integer n, $\mathrm{neg}(n) < 1/p(n)$. If S is a probability space, then the probability assignment $x \leftarrow_R S$ means that an element x is chosen at random according to S. If F is a finite set, then $x \leftarrow_R F$ denotes that x is chosen uniformly from F. If p is a predicate and $S_1, S_2, \ldots, S_m$ are probability spaces, then the notation $\mathtt{Pr}\left[x_1 \leftarrow_R S_1, x_2 \leftarrow_R S_2, \ldots\ x_m \leftarrow_R S_m : p(x_1, x_2, \cdots, x_m)\right]$ denotes the probability that $p(x_1, \cdots, x_m)$ will be true after the ordered execution of the probabilistic assignments $x_1 \leftarrow_R S_1, \ldots, x_m \leftarrow_R S_m$. Let A and B be interactive Turing machines, we use $(a \leftarrow A(\cdot) \leftrightarrow B(\cdot) \rightarrow b)$ to denote that a and b are two random variables corresponding to the outputs of A and B as a result of their joint computation.

We use the notation used by Camenisch and Stadler in [14] for the various zero-knowledge proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

$$\mathtt{PK}\{(\alpha, \beta) : y = g^\alpha h^\beta \wedge (u \le \alpha \le v)\}$$

denotes a zero-knowledge proof of knowledge of integers α and β, such that $y = g^\alpha h^\beta$ holds and $u \le \alpha \le v$.

2.2 Security assumptions

The security of our construction is based on the strong RSA assumption, which assumes that it is infeasible to solve the following problem: Given an RSA modulus n and a random $x \leftarrow_R \mathbb{Z}_n^*$, find $e > 1$ and $y \in \mathbb{Z}_n^*$ such that $y^e = x \bmod n$. The strong RSA was introduced by Barić and Pfitzmann [3] and has been used in proving the security of many cryptographic schemes (e.g., [19, 18, 16]). It can be formally stated as follows:

Assumption 1 (strong RSA assumption) For every probabilistic polynomial-time algorithms A,

$$\mathtt{Pr}\left[n \leftarrow G(1^k), x \leftarrow_R \mathbb{Z}_n, (y, e) \leftarrow A(n, x) : y^e = x \pmod n \wedge 1 < e < n\right] = \textit{neg}(k)$$

where $G(1^k)$ is a algorithm that generates a RSA modulus n of size k, and $\textit{neg}(k)$ is a negligible function.

Our security proofs also use the following lemma ([23, 16]):

Lemma 1. *For any integer n, given integers $u, v \in \mathbb{Z}_n^*$ and $a, b \in \mathbb{Z}$, such that $u^a = v^b \bmod n$ and $\gcd(a, b) = 1$, one can efficiently compute $x \in \mathbb{Z}_n^*$ such that $x^a = v \bmod n$.*

To see the correctness of this lemma, observe that, as $\gcd(a, b) = 1$, one can use the extended Euclidian algorithm to find $c, d \in \mathbb{Z}$ such that $bd = 1 + ac$. Let $x = (u^d v^{-c} \bmod n)$, then

$$x^a = u^{ad} v^{-ac} = (u^a)^d v^{-ac} = (v^b)^d v^{-ac} = v \pmod n.$$

3 Universal Accumulators

We now give a formal definition of universal accumulators and present our construction.

3.1 Definition of universal accumulators

Definition 1. Let k be a security parameter, a secure universal accumulator for a family of input $\{\mathcal{X}_k\}$ is a family of functions $\{\mathcal{F}_k\}$ with the following properties:

- *Efficient Generation*: There is an efficient probabilistic polynomial time algorithm G that on input 1^k produces a random function f of $\mathcal{F}_k$. Additionally, G also outputs some auxiliary information about f, denoted as aux_f.
- *Efficient Evaluation*: Each $f \in \mathcal{F}_k$ is a polynomial time function, on input $(g, x) \in \mathcal{G}_f \times \mathcal{X}_k$, outputs a value $h \in \mathcal{G}_f$, where $\mathcal{G}_f$ is the input domain for the function f, and $\mathcal{X}_k$ is the input domain for the elements to be accumulated.
- *Quasi-Commutative*: For all $f \in \mathcal{F}_k$, for all $g \in \mathcal{G}_f$, and for all $x_1, x_2 \in \mathcal{X}_k$, $f(f(g, x_1), x_2) = f(f(g, x_2), x_1)$. If $X = \{x_1, \ldots, x_m\} \subset \mathcal{X}_k$, we use $f(g, X)$ to denote $f(f(\ldots(g, x_1), \ldots), x_m)$.
- *Membership Witness*: For each $f \in \mathcal{F}_k$, there is a membership verification function ρ_1. Let $c \in \mathcal{G}_f$ and $x \in \mathcal{X}_k$. A value w_1 is called membership witness if $\rho_1(c, x, w_1) = 1$.
- *Nonmembership Witness*: For each $f \in \mathcal{F}_k$, there is a nonmembership verification function ρ_2. Let $c \in \mathcal{G}_f$ and $x \in \mathcal{X}_k$. A value w_2 is called nonmembership witness if $\rho_2(c, x, w_2) = 1$.
- *Security*: A universal accumulator scheme is secure if, for all probabilistic polynomial-time adversary A_k,

$$\mathtt{Pr}\begin{bmatrix} f \leftarrow G(1^k); g \leftarrow_R \mathcal{G}_f; (x, w_1, w_2, X) \leftarrow A_k(f, \mathcal{G}_f, g) : \\ x \in \mathcal{X}_k; X \subset \mathcal{X}_k; \rho_1(f(g, X), x, w_1) = 1; \rho_2(f(g, X), x, w_2) = 1 \end{bmatrix} = \mathrm{neg}(k)$$

  In other words, it is computationally infeasible to find both a valid membership witness and a valid nonmembership witness for any x in $\mathcal{X}_k$. Note that this is equivalent to say that, given any set $X \in \mathcal{X}_k$, it is computationally infeasible to find $x \in X$ with a valid nonmembership witness or find $x \in \mathcal{X}_k \backslash X$ with a valid membership witness.

The preceding definition is similar to the one of Camenisch and Lysyanskaya [10], the main difference is that our definition requires witnesses for nonmembership elements.

3.2 Our construction

Our construction builds upon the construction of Camenisch and Lysyanskaya [10]. The difference is that we give an efficient solution for nonmembership witness.

Construction 1 (Universal Accumulators) Let k be a security parameter. Let $\ell = \lfloor k/2 \rfloor - 2$. We use $\mathcal{X}_k$ to denote the set of all primes in $\mathbb{Z}_{2^\ell}$. $\mathcal{X}_k$ is the input domain for the elements to be accumulated.[4] We use $\mathcal{F}_k$ to denote the family of functions corresponding to safe-prime products of length k. The construction takes the following steps.

[4] As in [10], the input domain $\mathcal{X}_k$ has to be primes. If the required input domain is the set of all possible strings, we need to map arbitrary strings to prime numbers. A number of approaches for doing this have been proposed in the literature, see, e.g., [3, 19, 20].

– The generation algorithm G takes 1^k as input and outputs a random modulus n of length k that is a safe prime, that is, $n = pq$, where $p = 2p' + 1$, $q = 2q' + 1$, p and q have equal length, and p, q, p', q' are all prime number.
– We use f_n to denote the function corresponding to modulus n. The auxiliary information aux_f for f_n is the factorization of n. The input domain $\mathcal{G}_f$ for f_n is defined as $\mathcal{G}_f = \{g \in QR_n : g \neq 1\}$ where QR_n denotes the group of quadratic residues modulo n.
– For $f = f_n$, $f(g, x) = g^x \bmod n$.
– For $f = f_n$, the membership verification function ρ_1 is defined as $\rho_1(c, x, c_x) = 1$ if and only if $(c_x)^x = c$, where $c_x \in \mathcal{G}_f$ is the membership witness for x. The nonmembership verification function ρ_2 is defined as $\rho_2(c, x, a, d) = 1$ if and only if $c^a = d^x g \pmod{n}$, where $(a, d) \in \mathbb{Z}_{2^\ell} \times \mathcal{G}_f$ is the nonmembership witness for x.

It is easy to see that, given (g, x), we can efficiently compute $f(g, x)$. It is also easy to see that f is quasi-commutative, that is, $f(f(g, x_1), x_2) = f(f(g, x_2), x_1) = g^{x_1 x_2} \bmod n$. Note that membership and nonmembership witnesses can be computed with or without the auxiliary information. It is much more expensive to compute witness without the auxiliary information. In our application, we do not need to compute witness using the auxiliary information; but they may be useful in other settings.

One difference between our construction and Camenisch and Lysyanskaya's accumulator scheme [10] is that their construction does not require to publish g. In our construction, g needs to be public for nonmembership queries. Note that a party that knows g and the value of the accumulator, and suspects that $x_1, \ldots, x_n$ are the values used to compute the accumulator, can easily verify its guess. This is not really a problem because (1) it is not required to hide the accumulated values, and (2) it is easy to prevent this attack by adding a random value x_0 to the set of values used to compute the accumulator.

How to compute witness without the auxiliary information Suppose $X = \{x_1, \ldots, x_m\}$ is a subset of $\mathcal{X}_k$ and g is a random value in QR_n. Let u denote $\prod_{i=1}^m x_i$. By definition $f(g, X) = g^u \bmod n$. As in the previous accumulator schemes [4, 3, 10], for any $x \in X$, we can compute the membership witness for x as $c_x = g^{u/x} \bmod n$. To verify the witness, one checks that $x \in \mathcal{X}_k$ and $(c_x)^x = c \bmod n$.

For any $x \in \mathcal{X}_k \backslash X$, since $x, x_1, \ldots, x_m$ are distinct prime numbers, $\gcd(x, u) = 1$. We can find $a \in \mathbb{Z}_{2^\ell}$ and $b \in \mathbb{Z}$ such that $au + bx = 1$. The value a and b can be computed as follows: we first use Euclid algorithm to find a' and b' such that $a'u + b'x = 1$. As x is a positive integer in $\mathbb{Z}_{2^\ell}$, we can always find an integer k such that $a' + kx \in \mathbb{Z}_{2^\ell}$. Observe that $(a' + kx)u + (b' - ku)x = 1$, therefore $a = a' + kx$ and $b = b' - ku$. Let $d = g^{-b} \bmod n$, the nonmembership witness for x is (a, d). To verify the witness, one checks that $x \in \mathcal{X}_k$, $a \in \mathbb{Z}_{2^\ell}$, and $c^a = d^x g \pmod{n}$, which holds because $c^a = g^{ua} = g^{1-bx} = g^{-bx} g = d^x g \pmod{n}$.

How to compute witness with the auxiliary information The membership witness and nonmembership witness can be computed efficiently given the auxiliary information aux_f. Suppose there is a trusted group manager who knows aux_f, maintains the set X, and has already computed the accumulator $c = f(g, X)$, the group manager can compute (non)membership witness for any $x \in \mathcal{X}_k$ with *one* short modular exponentiation.

For $x \in X$, the group manager first checks whether $x \in X$, then computes $a = x^{-1} \bmod \phi(n)$, and finally computes $c_x = c^a \bmod n$. The membership witness for x is c_x. It is easy to verify the correctness of the witness as $(c_x)^x = (c^a)^x = c^{x^{-1} \cdot x \bmod \phi(n)} = c \pmod n$.

For $x \in \mathcal{X}_k \backslash X$, let $u' = u \bmod \phi(n)$, the group manager first checks whether $\gcd(x, u') = 1$.

– If $\gcd(x, u') = 1$, the group manager finds a and b such that $au' + bx = 1$, and sets the nonmembership witness for x as $(a, g^{-b} \bmod n)$. The nonmembership witness is correct because $c^a = (g^u)^a = (g^{u'})^a = g^{u'a} = g^{1-bx} = g^{-bx}g = d^x g \pmod n$.
– If $\gcd(x, u') \neq 1$, the group manager finds a and b such that $au + bx = 1$, then computes $b' = b \bmod \phi(n)$, and sets the nonmembership witness for x as $(a, g^{-b'} \bmod n)$. The nonmembership witness is correct because $c^a = g^{ua} = g^{1-bx} = (g^{-b})^x g = (g^{-b'})^x g = d^x g \pmod n$.

Observe that, the second case is slightly more expensive than the first case. The reason is that, in the second case, the group manager needs to find a and b such that $au + bx = 1$ where u could potentially be large, i.e., size linear to m. Yet, in either case, the exponent in the modular exponentiation computed by the group manager is smaller than $\phi(n)$. Thus the nonmembership witness can be calculated efficiently. Also observe that, the number of $x \in \mathcal{X}_k$ such that $\gcd(x, u') \neq 1$ is less than k, as x must be a prime.

Note that computing witness using auxiliary information may not apply to all scenarios. In some applications, it is not allowed to reveal the auxiliary information to the party who computes the accumulator, since the auxiliary data enables her to prove arbitrary statements. In the case when the party who computes the accumulator is trusted, it is acceptable to give her the auxiliary information.

Theorem 1. *Under the strong RSA assumption, the above construction is a secure universal accumulator.*

Proof. We assume all the arithmetic operations in this proof are modulo n unless specified otherwise. The strong RSA assumption says that given a RSA modulus n and a random value $g \leftarrow_R QR_n$, it is computationally infeasible to find x and y such that $x > 1$ and $y^x = g$.

Suppose there exists a polynomial time adversary $\mathcal{A}$, which on input n and $g \in QR_n$, outputs $c_x \in \mathcal{G}_f$, $d \in \mathcal{G}_f$, $x \in \mathcal{X}_\ell$, $a \in \mathbb{Z}_{2^\ell}$, and $X = \{x_1, \ldots, x_m\} \subset \mathcal{X}_\ell$, such that $c = g^{x_1 \cdots x_m}$, $(c_x)^x = c$, and $c^a = d^x g$. We can construct an algorithm $\mathcal{B}$ to break the strong RSA assumption by invoking $\mathcal{A}$.

Let u to denote $\prod_{i=1}^{m} x_i$. We consider two cases. In the first case, assume $x \in X$, the adversary can compute u, a, d, and x, such that $c = g^u$ and $c^a = d^x g$. That is, the adversary computes u, a, d, and x such that $g^{au-1} = d^x$. Because $x \in X$, $x \mid u$, and $\gcd(au - 1, x) = 1$. By lemma 1, we can efficiently find y such that $y^x = g$.

In the second case, assume $x \notin X$, the adversary can compute u, c_x, and x, such that $c = g^u$ and $(c_x)^x = c$. In other words, the adversary can find u, c_x, and x such that $g^u = (c_x)^x$. As $x_1, \ldots, x_m$ are all prime, and $x \notin X$, clearly $\gcd(x, u) = 1$. By lemma 1, we can efficiently find y such that $y^x = g$.

We now construct an efficient algorithm $\mathcal{B}$ that breaks the strong RSA assumption as follows. Given a RSA modulus n and $g \leftarrow_R QR_n$, $\mathcal{B}$ invokes $\mathcal{A}$ with input n and g, and obtains outputs c_x, d, x, a, X from $\mathcal{A}$. By the preceding arguments, $\mathcal{B}$ can efficiently compute y from c_x, d, x, a, X such that $y^x = d$, which contradicts to the strong RSA assumption. □

Corollary 1. *In the above construction, for any $f \in \mathcal{F}_k$ and any given set $X \subset \mathcal{X}_f$, it is computationally infeasible to find $x \in X$ with a valid nonmembership witness.*

Proof. This follows directly from Theorem 1. □

Note that, in our security definition of the universal accumulator, we limit the adversary to choose x only from $\mathcal{X}_k$. It is acceptable because when a user proves membership or nonmembership to a verifier, the verifier can first check whether $x \in \mathcal{X}_k$, if not, the verifier can reject the proof. If a user can prove that a value x was not accumulated in an accumulator in an anonymous fashion (see Section 5), Corollary 1 guarantees that x is not a member of the accumulated set X.

4 Dynamic Universal Accumulators

Camenisch and Lysyanskaya [10] proposed the concept of dynamic accumulators in which one can dynamically add and delete elements. In this section, we first give the definition of dynamic universal accumulators, then present a dynamic universal accumulator based on our construction in the previous subsection.

4.1 Definition of dynamic universal accumulators

Definition 2. A universal accumulator is *dynamic* if it has the following properties:

- *Efficient Update of Accumulator* There exists an efficient algorithm D such that, suppose $c = f(g, X)$, if $\hat{x} \notin X$, then $D(c, \hat{x}) = \hat{c}$ such that $\hat{c} = f(g, X \cup \{\hat{x}\})$; if $\hat{x} \in X$, then $D(\mathsf{aux}_f, c, \hat{x}) = \hat{c}$ such that $\hat{c} = f(g, X \backslash \{\hat{x}\})$.
- *Efficient Update of Membership Witness* Let c and $\hat{c}$ be the original and updated accumulators respectively and $\hat{x}$ be the new updated element. There exists an efficient algorithm W_1 such that, if $x \neq \hat{x}$, $x \in X$, and $\rho_1(c, x, w) = 1$, then $W_1(w, c, \hat{c}, x, \hat{x}) = \hat{w}$ such that $\rho_1(\hat{c}, x, \hat{w}) = 1$.
- *Efficient Update of Nonmembership Witness* Let c and $\hat{c}$ be the original and updated accumulators respectively and $\hat{x}$ be the new updated element. There exists an efficient algorithm W_2 such that, if $x \neq \hat{x}$, $x \notin X$, and $\rho_2(c, x, w) = 1$, then $W_2(w, c, \hat{c}, x, \hat{x}) = \hat{w}$ such that $\rho_2(\hat{c}, x, \hat{w}) = 1$.

Of course, it is easy to perform updates using computations that are linear in the size of the accumulated set X, i.e., compute the witnesses from the scratch. In the above definition, the term "efficient" means that the time complexity of each update operation is independent of the size of X. Note that, update of a membership witness or a nonmembership witness is achieved without the auxiliary information. That is, given the original and new accumulators, one can update its witness locally. This is a very useful feature: Suppose the group manager updates the set X, computes the new accumulator, and broadcasts its value to all users. Each user can update her witness *locally* without any help from the group manager.

In terms of security requirement, loosely speaking, a dynamic universal accumulator is secure against an adaptive adversary if the adversary cannot win the following game. Suppose a group manager sets up the function f and the value g and hides the auxiliary information aux_f. The adversary adaptively modifies the set X. Whenever a value x is inserted into or deleted from X, the manager calls algorithm D and publishes the updated accumulator. In the end, the adversary outputs $\hat{x} \notin X$ and a valid membership witness for $\hat{x}$ or outputs $\hat{x} \in X$ and a valid nonmembership witness for $\hat{x}$. The formal security definition of dynamic universal accumulator is stated as follows.

Definition 3. Let $\{\mathcal{F}_k\}$ be the family of universal accumulator functions defined in Definition 1. Let M be an interactive Turing machine that receives input (f, aux_f, g), where $f \in \mathcal{F}_k$, aux_f is the auxiliary information about f, and $g \in \mathcal{G}_f$. M maintains a list of values X which is initially empty. The initial accumulator c is set to be g. M responds to two types of messages: for message (add, x), it makes sure that $x \in \mathcal{X}_k$, adds x to the set X, modifies c by running D, and then sends back the updated c; for message (delete, x), it checks that $x \in X$, deletes it from the set X, updates c by running D, and sends back the updated c. In the end, M outputs the current values for X and c. A dynamic universal accumulator scheme is *secure* if, for all probabilistic polynomial-time adversary A_k,

$$\mathtt{Pr}\left[\begin{array}{l} f \leftarrow G(1^k); g \leftarrow_R \mathcal{G}_f; \\ (x, w_1, w_2, X) \leftarrow A_k(f, \mathcal{G}_f, g) \leftrightarrow M(f, \mathsf{aux}_f, g) \rightarrow (X, c): \\ x \in \mathcal{X}_k; X \subset \mathcal{X}_k; c = f(g, X); \rho_1(c, x, w_1) = 1; \rho_2(c, x, w_2) = 1 \end{array}\right] = \text{neg}(k)$$

We now show that if a secure universal accumulator is dynamic under Definition 2, then this dynamic universal accumulator is secure against adaptive adversaries.

Remark 1. A dynamic universal accumulator is secure against an adaptive adversary if the underlying universal accumulator is secure.

The above remark is straight-forward using reduction argument. We can show that if an adversary $\mathcal{A}$ breaks the security property in Definition 3, we could build another adversary $\mathcal{B}$ to break the security property of a universal accumulator in Definition 1 by invoking $\mathcal{A}$. On input $(f, \mathcal{G}_f, g)$, $\mathcal{B}$ passes these values to $\mathcal{A}$. Because $\mathcal{A}$ needs to interacts with the manager M for updating elements, we let $\mathcal{B}$ act as the manager: if $\mathcal{A}$ sends an (add, x) query, $\mathcal{B}$ simply inserts x into X and computes $c = f(g, X)$; if $\mathcal{A}$ sends a (delete, x) query, $\mathcal{B}$ removes x from X and computes $c = f(g, X)$. In the end, if $\mathcal{A}$ outputs an element $x \in \mathcal{X}_k$ with a valid membership witness w_1 and a valid nonmembership witness w_2, $\mathcal{B}$ outputs (x, X, w_1, w_2). Clearly, $\mathcal{B}$ breaks the security property in Definition 1.

4.2 Our construction

Construction 2 (Dynamic Universal Accumulators) Our construction is built on Construction 1 with the following additional functionalities:

- *Update of Accumulator:* Adding a value $\hat{x}$ to the accumulator c can be computed as $\hat{c} = c^{\hat{x}} \bmod n$. Deleting a value $\hat{x}$ from the accumulator is computed as $\hat{c} = D(\phi(n), c, \hat{x}) = c^{\hat{x}^{-1} \bmod \phi(n)} \bmod n$, where $\phi(n)$ is the auxiliary information.
- *Update of Membership Witness:* Let w be the original membership witness of x. Let c and $\hat{c}$ be the original and new accumulators, respectively. This construction is the same as the one in [10].

  1. *Addition*. Suppose $\hat{x}$ has been added, the new membership witness can be computed as $\hat{w} = f(w, \hat{x}) = w^{\hat{x}} \bmod n$. It is easy to verify that $\rho_1(\hat{c}, x, \hat{w}) = 1$.
  2. *Deletion*. Suppose $\hat{x} \neq x$ has been deleted, the new membership witness $\hat{w}$ can be computed as follows. Algorithm W_1 chooses two integer a and b such that $ax + b\hat{x} = 1$ and then $\hat{w} = w^b \hat{c}^a \bmod n$. We can verify that:

  $$\hat{w}^x = (w^b \hat{c}^a)^x = ((w^b \hat{c}^a)^{x\hat{x}})^{1/\hat{x}} = (c^{b\hat{x}} c^{ax})^{1/\hat{x}} = \hat{c} \pmod n$$

- *Update of Nonmembership Witness:* Let (a, d) be the original nonmembership witness of x.

  1. *Addition*. Suppose $\hat{x} \neq x$ has been added, given $c, \hat{c}, x, \hat{x}$ such that $\hat{c} = c^{\hat{x}} \bmod n$, the new nonmembership witness $(\hat{a}, \hat{d})$ can be computed as follows. Algorithm W_2 first finds two integers $\hat{a}_0$ and r_0 such that $\hat{a}_0\hat{x} + r_0 x = 1$. It is easy to find such $\hat{a}_0$ and r_0 because $\hat{x}$ and x are distinct primes. Multiplying by a to both side of the above equation, we have $\hat{a}_0 a\hat{x} + r_0 ax = a$. W_2 then computes $\hat{a} = \hat{a}_0 a \bmod x$, and find $r \in \mathbb{Z}$ such that $\hat{a}\hat{x} = a + rx$. Note that $\hat{a} \in \mathbb{Z}_{2^\ell}$. In the end, W_2 computes $\hat{d} = dc^r \bmod n$. We can verify $\rho_2(\hat{c}, x, \hat{a}, \hat{d}) = 1$, or, $\hat{c}^{\hat{a}} = \hat{d}^x g$ holds:

  $$\hat{c}^{\hat{a}} = c^{\hat{a}\hat{x}} = c^{a+rx} = c^{rx} c^a = c^{rx} d^x g = (dc^r)^x g = \hat{d}^x g \pmod n$$

  2. *Deletion*. Suppose $\hat{x}$ has been deleted, given $c, \hat{c}, x, \hat{x}$ such that $c = \hat{c}^{\hat{x}} \bmod n$, the new nonmembership witness $(\hat{a}, \hat{d})$ can be computed as follows. Algorithm W_2 chooses an integer r such that $a\hat{x} - rx \in \mathbb{Z}_{2^\ell}$ (there always exists such r because $x \in \mathbb{Z}^*_{2^\ell}$), then let $\hat{a} = a\hat{x} - rx$ and $\hat{d} = d\hat{c}^{-r} \bmod n$. We can verify $\rho_2(\hat{c}, x, \hat{a}, \hat{d}) = 1$, or, in other words, $\hat{c}^{\hat{a}} = \hat{d}^x g$ holds:

  $$\hat{c}^{\hat{a}} = \hat{c}^{a\hat{x}-rx} = c^a \hat{c}^{-rx} = d^x g \hat{c}^{-rx} = (d(\hat{c})^{-r})^x g = \hat{d}^x g \pmod n$$

Note that we can add or delete several values together simply by letting $\hat{x}$ be the product of the added or deleted values. This is also true for updating (non)membership witness.

5 Efficient Proof That a Committed Value Was Not Accumulated

We now present a useful building block for certificate revocation in an anonymous setting – a protocol that proves a committed value was not accumulated in the accumulator. Suppose that a group manager compiles a list of revoked users and publishes the accumulator for the set of revoked serial numbers[5]. If a regular user wants to prove that she is not in the revocation list, she simply shows her serial number and the corresponding nonmembership witness. However, such approach reveals the user's serial number (thus the identity as well). The building block presented in this section enables such nonmembership proof in an anonymous fashion, i.e., without revealing the serial number. The idea here is that the user first commits her serial number in her certificate, then proves that the committed serial number was not accumulated in the revocation list. We shall describe in details how this building block is used for certificate and membership revocation in the anonymous setting in the next section.

The commitment scheme that we use in this section is developed by Fujisaki and Okamoto [18] and improved by Damgård and Fujisaki [17]. The parameters of the this commitment scheme are (n_1, g_1, h_1), where n_1 is a special RSA modulus of length k_1, h_1 is a random value in QR_{n_1}, and g_1 is a random value in the group generated by h_1. To commit a value x, the committer chooses a random $r \leftarrow_R \mathbb{Z}_{n_1}$ and computes the commitment $\mathsf{commit}(x, r) = g_1^x h_1^r \bmod n_1$. The Fujisaki and Okamoto's commitment scheme is statistically hiding and computationally binding if factoring is hard. Note that the protocol described next could also work for other commitment schemes, such as the Pedersen commitment [22], with only minor modifications.

For our protocol, we require an element h in QR_n such that $\log_g h$ is unknown to the prover, where g and n are the parameters of the universal accumulators described in the previous sections. To prove that given a commitment c_1 and an accumulator c, the value committed in c_1 has not been accumulated in c, we build the following zero-knowledge proof protocol. The common inputs to the protocol are $c_1, n_1, g_1, h_1, c, n, g$, and h. The prover has additional inputs: x, r, a, d such that $c_1 = g_1^x h_1^r \bmod n_1$ and $c^a = d^x g \bmod n$, where the first equation shows that x is the committed value of c_1 and the second equation shows that x was not accumulated in c.

Protocol 1 $\mathtt{PK}\{(x, r, a, d) : c_1 = g_1^x h_1^r \ \wedge\ c^a = d^x g \ \wedge\ x \in \mathbb{Z}_{2^\ell} \ \wedge\ a \in \mathbb{Z}_{2^\ell}\}$

1. The prover chooses, uniformly at random, values w, r_x, r_a, r_w, r_z, and r_e of length k. The prover computes the following values (modulo n): $c_x = g^x h^{r_x}$, $c_a = g^a h^{r_a}$, $c_d = dg^w$, $c_w = g^w h^{r_w}$, $z = xw$, $c_z = g^z h^{r_z}$, and $c_e = (c_d)^x h^{r_e}$. The prover sends $(c_x, c_a, c_d, c_w, c_z, c_e)$ to the verifier and carry out the following zero-knowledge proofs of knowledge.
   Note that $c_e = (c_d)^x h^{r_e} = (dg^w)^x h^{r_e} = d^x g^{xw} h^{r_e} = g^{-1} c^a g^z h^{r_e}$.
2. The prover proves to the verifier that the value committed in c_1 in bases (g_1, h_1) is the same as the value committed in c_x in bases (g, h):

$$\mathtt{PK}\{(\varepsilon, \rho, \rho_x) : c_1 = g_1^\varepsilon h_1^\rho \bmod n_1 \ \wedge\ c_x = g^\varepsilon h^{\rho_x} \bmod n\}$$

[5] We assume that each group member in the system has a unique prime serial number.

3. The prover proves to the verifier that the value committed in c_e in bases (c_d, h) is the same as the value committed in c_x:

$$\mathtt{PK}\{(\varepsilon, \rho_e, \rho_x) : c_e = (c_d)^{\varepsilon} h^{\rho_e} \ \wedge \ c_x = g^{\varepsilon} h^{\rho_x}\}$$

4. The prover proves to the verifier that $c_e g$ is also a commitment in bases $((c, g), h)$, and the values committed in $c_e g$ are the same as the values committed in c_a, c_z, and the power of h in $c_e g$ is the same as in c_e in bases (c_d, h):

$$\mathtt{PK}\{(\sigma, \tau, \varepsilon, \rho_a, \rho_z, \rho_e) : \\ c_e g = c^{\sigma} g^{\tau} h^{\rho_e} \ \wedge \ c_a = g^{\sigma} h^{\rho_a} \ \wedge \ c_z = g^{\tau} h^{\rho_z} \ \wedge \ c_e = (c_d)^{\varepsilon} h^{\rho_e}\}$$

5. The prover proves to the verifier that c_z is a commitment to the product of values committed in c_x and c_w:

$$\mathtt{PK}\{(\sigma, \tau, \varepsilon, \rho_z, \rho_w, \rho_x, \rho) : \\ c_z = g^{\tau} h^{\rho_z} \ \wedge \ c_w = g^{\sigma} h^{\rho_w} \ \wedge \ c_x = g^{\varepsilon} h^{\rho_x} \ \wedge \ c_z = (c_w)^{\varepsilon} h^{\rho}\}$$

6. The prover proves to the verifier that c_x is a commitment to an integer of length ℓ, and that c_a is a commitment to an integer of length ℓ:

$$\mathtt{PK}\{(\varepsilon, \sigma, \rho_x, \rho_a) : c_x = g^{\varepsilon} h^{\rho_x} \ \wedge \ c_a = g^{\sigma} h^{\rho_a} \ \wedge \ \varepsilon \in \mathbb{Z}_{2^{\ell}} \ \wedge \ \sigma \in \mathbb{Z}_{2^{\ell}}\}$$

Note that the preceding protocol is similar to the one proposed by Camenisch and Lysyanskaya [11]. The zero-knowledge proof protocol in [11] is used to prove knowledge of a signature, whereas our protocol is to prove knowledge of a nonmembership witness. The details of the zero-knowledge proof protocols in each step are omitted, as these zero-knowledge proof protocols are standard in the literature, e.g., a protocol for proving knowledge of equality of representation modulo two composite modulus in step 2 , 3, and 4 can be found in [13], a protocol for zero-knowledge proof that a committed value is the product of two other committed values in step 5 can be found in [12, 17], a protocol for proving that a committed value lies in a given range in step 6 can be found in [6].

Theorem 2. *The preceding protocol is a zero-knowledge proof of knowledge of the values* (x, r, a, d) *such that* $c_1 = g_1^x h_1^r \bmod n_1$ *and* (a, d) *is a valid nonmembership witness of* x *for accumulator* c.

Proof. The completeness property of Protocol 1 is obvious. The zero-knowledge property of Protocol 1 is also clear. The simulator first computes the commitments c_x, c_a, c_d, c_w, c_z, and c_e at random. Then the simulator invokes the simulator for the zero-knowledge proofs of knowledge of each step. Because the commitments reveal nothing statistically and the proofs of knowledge protocols at each step are zero-knowledge, the preceding protocol is zero-knowledge.

We now show that there exists a knowledge extractor that outputs a valid committed value x and a valid nonmembership witness for x. Our extractor will invoke the extractor for the zero-knowledge proof protocols at each step as a building block. If our extractor fails, then we are able to set up a reduction to break the strong RSA assumption. Suppose the extractor succeeds and computes $(x, a, w, z, r, r_x, r_a, r_w, r_z, r_e)$ such that

$$c_1 = g_1^x h_1^r \tag{1}$$
$$c_x = g^x h^{r_x} \tag{2}$$
$$c_e = (c_d)^x h^{r_e} \tag{3}$$
$$c_a = g^a h^{r_a} \tag{4}$$
$$c_z = g^z h^{r_z} \tag{5}$$
$$c_e g = c^a g^z h^{r_e} \tag{6}$$
$$c_z = g^{xw} h^{r_z} \tag{7}$$

where the equations (1) and (2) come from the extractor in step 2 of the protocol, the equations (2) and (3) come from the extractor in step 3 of the protocol, the equation (4), (5), and (6) come from the extractor in step 4 of the protocol, the equation (5) and (7) come from the extractor in step 5 of the protocol. From equations (5) and (7), we get $z = xw$. Equations (3) and (6) imply that $(c_d)^x g = c^a g^z$. Let $d = c_d/g^w$, we have $d^x = (c_d/g^w)^x = c_d^x/g^z = c^a g^{-1}$, or equivalently, $c^a = d^x g$. Since we also know that x and a are of length ℓ, we can output (x, r, a, d) such that c_1 is a commitment of x, and (a, d) is a valid nonmembership witness for x. □

6 Application to Certificate and Membership Revocation

In this section, we show that the dynamic universal accumulator we constructed can be used for efficient membership revocation for group signatures, anonymous credentials, and direct anonymous attestation schemes; right after a brief review of these schemes.

6.1 Review of group signatures, anonymous credentials, and direct anonymous attestation

Group signatures, first introduced by Chaum and van Heyst [15], provide anonymity for signers. In a group signature scheme, each group member can sign messages such that the resulting signatures do not reveal the identity of the signer. A number of group signature schemes have been proposed, e.g., [1, 2, 5, 14]. Formally speaking, a group-signature scheme with membership revocation is a digital signature scheme comprised of the following procedures:

– *Setup:* On input a security parameter, this probabilistic algorithm outputs the initial group public key and the secret key for the group manager.
– *Join:* A protocol between the group manager and a user that results in the user becoming a new group member. The user's output is a membership certificate and a membership secret.
– *Sign:* A probabilistic algorithm that on input a group public key, a membership certificate, a membership secret, and a message m outputs group signature of m.

– *Verify:* An algorithm for establishing the validity of a group signature of a message with respect to a group public key.
– *Open:* An algorithm that, given a message, a valid group signature on it, a group public key and a group managers secret key, determines the identity of the signer.
– *Revocation:* An algorithm for the group manager to remove a member from the group. This algorithm results in an updated group's public key and some other information to be stored in a public server.
– *Membership Update:* An algorithm for the users to update their membership certificates and membership secrets using the information available in the public server and the current group public key.

A secure group signature scheme must satisfy the *anonymity* and *unlinkability* property. The anonymity property says that, given a valid signature of some message, identifying the actual signer is computationally hard for everyone but the group manager. The unlinkability property means that deciding whether two different valid signatures were computed by the same group member is computationally hard.

In an anonymous credential system [21, 9, 11], a user can demonstrate to a verifier that she has a credential, but the verifier cannot infer anything about who the user is other than the fact that the user has the right credential. The Camenisch and Lysyanskaya [9, 11] credential system has a similar construction to the group signature schemes. Essentially, their system enables a credential holder to prove to a verifier that the credential holder has a signature signed by the certificate authority.

Direct Anonymous Attestation (DAA) was first proposed by Brickell, Camenisch, and Chen [7]. DAA enables remote authentication of a Trust Platform Module (TPM), while preserving the privacy of the user of the platform that contains the module. The DAA scheme can be seen as a group signature without the feature that a signature can be opened. The DAA scheme presented in [7] is similar to the signature scheme proposed in [11].

6.2 Incorporating revocation into group signature schemes

In this subsection, we use the group signatures scheme developed by Ateniese et al. [1] as an example, and show that our universal accumulator scheme can be integrated into the group signatures scheme to enable efficient revocation.

– *Setup:* In [1], the group manager chooses two ranges Γ and Λ and chooses $n = pq$ where p and q are safe primes. The group manager also picks $a, a_0, g, h \in QR_n$, and chooses a secret elements x and computes $y = g^x \bmod n$. The group public key is (n, a, a_0, g, h, y) and the secret key is (p, q, x). In addition, the group manager creates n' for the universal accumulators described in section 3, such that $\Gamma \subseteq \mathcal{X}_k$. The group manager also chooses a random $g' \in QR_{n'}$ and publishes (n', g') as the public parameters of the universal accumulator.
– *Join:* In [1], a user interacts with the group manager. In the end, the user obtains $e_i \in \Gamma$, $x_i \in \Lambda$, and A_i such that $a^{x_i} a_0 \equiv A_i^{e_i} \bmod n$. In addition, given the current revocation list $\{e_1, \ldots, e_m\}$, the group manager computes the nonmembership witness for e_i, and sends the witness to the user.

– *Sign and Verify:* In [1], the prover with private key (A_i, e_i, x_i) proves to the verifier that she is a member of the group. More specifically, the prover uses zero-knowledge proof of knowledge to prove the knowledge of (A_i, e_i, x_i) such that $a^{x_i} a_0 \equiv A_i^{e_i} \mod n$. In addition, the prover proves to the verifier that e_i is not in the revocation list. This can be done using the zero-knowledge proof protocol in Section 5. That is, the prover can first commit e_i and then prove that (1) the value committed is the same as the value in her private key, and (2) the value committed has not been accumulated in $\{e_1, \ldots, e_m\}$.
– *Revocation:* To revoke a member with private key (A_i, e_i, x_i) from the group, the group manager inserts e_i into the revocation list. Let c be the current accumulator. The group member updates $c = f(c, e_i) = c^{e_i} \mod n'$.
– *Membership Update:* Let $\hat{c}$ be the current accumulator and c be previous accumulator stored in the public server. Each group member updates her nonmembership witness accordingly using $\hat{c}$ and c using the algorithms presented in Section 4.

Analogously, we could integrate our revocation scheme using universal accumulators with other group signature schemes [2, 5, 14], anonymous credentials schemes [9, 11], compact e-cash scheme [8], and DAA scheme [7] with minor modifications. Observe that the group manager could generate both a valid membership witness and a valid nonmembership witness for a given group member. In our scheme, we assume that the group manager is trusted and it will not generate nonmembership witnesses for the members that are already in the revocation list.

7 Conclusion

We proposed a new cryptographic scheme called universal accumulators which enables one to condense to a set of values using a short accumulator, to efficiently compute a witness of the membership of any value that has been accumulated, and to efficiently compute a witness of the nonmembership of any value that has not been accumulated. We gave a construction for universal accumulators and proved its security based on the strong RSA assumption. We then presented a construction for dynamic universal accumulators in which one can add (or delete) inputs into (or from) the accumulated set with constant cost. Dynamic universal accumulators can be used for efficient membership revocation in the anonymous setting. Universal accumulators may be of independent interest in other applications as well.

Acknowledgement

Most of this work was done while Jiangtao Li and Rui Xue were at Purdue University. This work is supported by NSF IIS-0430274, NSF CCR-0325951, and sponsors of CERIAS. Rui Xue is also supported by the Fund of the China Scholarship Council and the NSFC Grant 60373048. We thank the anonymous reviewers for their helpful comments.

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology – CRYPTO '00*, pages 255–270, 2000.
2. G. Ateniese, D. Song, and G. Tsudik. Quasi-efficient revocation of group signatures. In *Proceedings of Financial Cryptography*, pages 183–197, 2001.
3. N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology – EUROCRYPT '97*, pages 480–494, 1997.
4. J. C. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Advances in Cryptology – EUROCRYPT '93*, pages 274–285, 1994.
5. D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *Proceedings of the 11th ACM conference on Computer and Communications Security (CCS)*, pages 168–177, 2004.
6. F. Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology – EUROCRYPT '00*, pages 431–444, May 2000.
7. E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM conference on Computer and Communications Security (CCS)*, pages 132–145, 2004.
8. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In *Advances in Cryptology – EUROCRYPT '05*, pages 302–321, 2005.
9. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology – EUROCRYPT '01*, pages 93–118, 2001.
10. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology – CRYPTO '02*, pages 61–76, 2002.
11. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *Proceedings of the 3rd Conference on Security in Communication Networks*, pages 268–289, 2002.
12. J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *In Advances in Cryptology – EUROCRYPT '99*, pages 106–121, 1999.
13. J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In *In Advances in Cryptology – CRYPTO '99*, pages 413–430, 1999.
14. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology – CRYPTO '97*, pages 410–424, 1997.
15. D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology – EUROCRYPT '91*, pages 257–265, Apr. 1991.
16. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS)*, pages 46–51, Nov. 1999.
17. I. Damgård and E. Fujisaki. An integer commitment scheme based on groups with hidden order. In *Advances in Cryptology: ASIACRYPT '02*, pages 125–142. Springer, 2002.
18. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology – CRYPTO '97*, pages 16–30, Aug. 1997.
19. R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology – EUROCRYPT '99*, pages 123–139, 1999.
20. M. T. Goodrich, R. Tamassia, and J. Hasic. An efficient dynamic and distributed cryptographic accumulator. In *Proceedings of the 5th International Conference on Information Security*, pages 372–388, 2002.

21. A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography, 6th Annual International Workshop, SAC '99*, pages 184–199, 1999.
22. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO '91*, pages 129–140, 1992.
23. A. Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Transactions on Computer Systems*, 1(1):38, Feb. 1983.