Efficient Accumulators Without Trapdoor Extended Abstract

Tomas Sander

International Computer Science Institute
1947 Center Street, Berkeley, CA 94704, USA
sander@icsi.berkeley.edu

Abstract. In 1994 Benaloh and de Mare introduced the notion of one way accumulators that allow to construct efficient protocols for proving membership in a list and related problems like time stamping and authentication. As required by Benaloh et al. unlike in signature based protocols no central trusted authority is (should be) needed. Accumulator based protocols do further improve on hash tree based protocols for proving membership in a list as verification and storage requirements are independent of the number of items in the list. Benaloh's et al. accumulator construction was based on exponentiation modulo a RSA modulus $N = PQ$.
As already noted by Benaloh et al. the party (or parties) who generated the needed RSA modulus N during system set up knows a factorization of N. This knowledge allows this party to completely bypass the security of accumulator based protocols. For example a time stamping agency could forge time stamps for arbitrary documents.
Thus these parties need to be trusted in (at least) two ways. First that they do not abuse their knowledge of the trapdoor and secondly to have had adequate security in place during system set up, which prevented outside attackers from getting hold of P and Q.
In this paper we describe a way to construct (generalized) RSA moduli of factorization unknown to anybody. This yields (theoretically) efficient accumulators such that "nobody knows a trapdoor" and the two above mentioned trust requirements in the parties who set up the system can be removed.
Keywords: one way accumulator, RSA modulus.

1 Introduction

The basic idea of the accumulator based authentication scheme suggested by Benaloh and de Mare in [2] is as follows: Let N be a RSA modulus and let $x \in (\mathbb{Z}/N\mathbb{Z})^\times$. Further let $\mathcal{L} = \{y_1, \ldots y_m\} \subset (\mathbb{Z}/N\mathbb{Z})^\times$ be a list of items for which membership to $\mathcal{L}$ should later be proved. The "accumulated hash value" z of the list $\mathcal{L}$ is defined to be the value $z = x^{y_1 \cdot y_2 \cdots y_m} \bmod N$. Assume now that Victor has obtained z over an authenticated channel. To prove membership of an element y to $\mathcal{L}$ Alice presents to Victor a y'th root r of z. Victor accepts if $r^y = z$.

Assume y in fact belongs to the list, i.e. let $y = y_i$. Then a y'th root of z can be computed by raising x to the power $a = \prod_{y \in \mathcal{L} \setminus \{y_i\}} y$. Thus if Alice has obtained a or computed it herself using her knowledge of the other items in $\mathcal{L}$ she can present x^a to Victor to prove that y contributed to the accumulated hash z, i.e. that $y \in \mathcal{L}$.

Barić and Pfitzmann [1] showed how this basic scheme can be turned into a provably secure scheme: assuming (variants of) the RSA assumption they describe an accumulator based protocol such that forging membership proofs is unfeasible. We postpone the discussion of the security of accumulator based protocols to Section 3.

A main advantage of accumulator based protocols for proving membership in a list over protocols using Merkle's authentication trees [7] is improved efficiency: in hash tree based protocols the elements of the list $\mathcal{L}$ to be authenticated are inserted as leaves of a hash tree. Assume Victor has obtained the root R of the hash tree over an authenticated channel. To prove membership of y to $\mathcal{L}$ Alice presents to Victor a hash path leading from the leaf y to the root R and Victor verifies the correctness of this path. Thus Alice needs to store $\log \sharp \mathcal{L}$ many values and Victor's work during checking the correctness of the path also depends logarithmically on the number of items $\mathcal{L}$. In contrast to this the complexity of accumulator based protocols is independent of the number of items hashed together: the length of a hash chain that Alice needs to present to prove membership to $\mathcal{L}$ collapses to one.

Although accumulator based schemes are more efficient they also have a serious disadvantage over hash tree based authentication schemes. Hash tree based schemes are secure under the sole cryptographic assumption that the hash function is strongly collision resistant: *no party* is able to forge membership proofs. This is no longer the case in accumulator based protocols. All efficient accumulator constructions known today are based on RSA type functions. The problem is that the RSA modulus N needs to be *constructed* during system setup. RSA moduli are typically found by choosing two random large primes P and Q and multiplying them together. Thus the party who provides N during system setup, knows also the "trapdoor" P and Q. [1] Unfortunately a party who knows P and Q can completely bypass the system's security. It can forge membership proofs for arbitrary values $y \in (\mathbb{Z}/N\mathbb{Z})^\times$. To do this the party computes a t s.t. $ty \equiv 1 \bmod \varphi(N)$. Now z^t is a y'th root of z that "proves" to Victor that $y \in \mathcal{L}$.

This is an undesirable property in many applications and already Benaloh et al. asked in their original paper [2] if accumulators "without trapdoor" could be constructed. The main result of this paper is that this question has an affirmative answer.

[1] Even if secure multiparty computation protocols are employed during system setup still a colluding curious coalition of this group could recover P and Q. Thus the general problem that someone "knows the trapdoor" remains also in the distributed setting.

Let us first describe two examples to illustrate weaknesses of accumulators that have a (known) trapdoor. In [2] an efficient round based time stamping protocol using accumulators was suggested. In each round the submitted documents are hashed together using the accumulator. Now the time stamping agency (or the parties that provided the modulus N) can forge arbitrary time stamps for any document of their choice. Thus the time stamping agency needs to be fully trusted to not abuse its knowledge of the trapdoor or to have it destroyed after system set up.

A different example is provided by an electronic cash scheme that was recently suggested by Sander and Ta-Shma in [9]. This system offers payers unconditional anonymity but unlike previous (blind signature based) schemes it is not vulnerable to the blackmailing [11] and the bank robbery attack. The serious bank robbery attack was described by Jakobsson and Yung in [6]. In this attack the secret key which the bank uses to sign coins is compromised, e.g. by an insider attack. In the scheme described in [9] users submit during withdrawal hash values of electronic coins to the bank which inserts them as leaves into a hash tree. Its root R is distributed to merchants by the bank. During payment a user demonstrates to the merchant the validity of a coin C by proving with a zero knowledge argument that there is a hash path from C to R.

This system is resistant to the bank robbery attack as there are no critical bank secrets that allow to forge money. To speed up the system it is asked in [9] whether accumulators without trapdoor exist that could substitute the hash trees. As remarked in [9] parties that know the trapdoor of an accumulator can forge arbitrary amounts of electronic money. Thus the resistance of the scheme against the bank robbery attack would at least partially be lost if existing accumulators were used. At least during system set up the scheme would again be vulnerable to the bank robbery attack. These examples motivate why it is useful to have accumulators without trapdoor.

1.1 Accumulators without trapdoor

A construction of an accumulator that does not have a trapdoor was given by Nyberg in [8]. This accumulator is provably secure. However the accumulator is not space efficient (and thereby also not time efficient): the accumulated hash of N items has length $N \log(N)$. Thus the potential performance advantages offered by accumulator based protocols are not preserved.

All efficient accumulators that have been found so far are based on exponentiation modulo a RSA modulus. Note that the RSA type accumulator described by Benaloh et al. would not have an (exploitable) trapdoor if we had an algorithm that outputs RSA moduli $N = PQ$ such the parties executing the algorithm had no way to extract P and Q. To the best of our knowledge no such algorithm is known today. Unfortunately we can not present such an algorithm here either.

We avoid the problem of constructing such RSA moduli and suggest to use "generalized RSA moduli **o**f **u**nknown complete **f**actorization " instead for the construction of accumulators. By abuse of notation we call these numbers RSA-UFOs. We call a number N a RSA-UFO if N has at least two large prime factors

P and Q such that it is infeasible for *any* coalition of players including those that generated N to find a splitting of N into factors N_1, N_2, such that $P \mid N_1$ and $Q \mid N_2$, i.e. it is infeasible to construct a splitting of N into factors that "separate" P and Q.

In this paper we suggest a (theoretically) efficient algorithm that constructs a RSA-UFO with very high probability: given a security parameter k and an element ϵ there is an algorithm with running time polynomial in k and $\log(1/\epsilon)$ that outputs an element that is a RSA-UFO w.r.t. two primes P and Q of bit length at least k with probability $1-\epsilon$. We call such a number a (k,ϵ)-RSA-UFO. By choosing ϵ sufficiently small any practically desirable degree of certainty can be achieved. The algorithm that outputs (k,ϵ)-RSA-UFOs is described in Section 2.

Using the techniques of Barić and Pfitzmann [1] we construct in Section 3 an accumulator using (k,ϵ)-RSA-UFOs such that "nobody knows a trapdoor". We show that this accumulator is provably secure under a natural RSA type assumption.

2 Constructing Generalized RSA Moduli Of Unknown Factorization

Our general strategy for the construction of (k,ϵ)-RSA-UFOs is as follows: We first prove that a constant fraction of integers with bit length (roughly) $\leq 3k$ has at least two distinct prime factors of bit length $\geq k$. Now one picks uniformly and independently $r = O(\log \frac{1}{\epsilon})$ integers $a_1, a_2, \ldots, a_r$ of bit length $\leq 3k$. The probability that at least one of these numbers has two distinct prime factors of length at least k is $\geq 1-\epsilon$. Thus the product $A := a_1 \cdot a_2 \cdot \ldots \cdot a_r$ has now at least one factor a_i which has two large distinct prime divisors P and Q w.v.h.p.. To avoid cheating the random choices should have been made by choosing the bits in the expansion of the numbers according to a public or publicly verifiable random source (see Paragraph 2.1 for a brief discussion on ways how this could be achieved).

We argue now that it is reasonable to believe that A is a (k,ϵ)-RSA-UFO. W. v. h. p. one of the factors, let's say w.l.o.g. a_1, has at least two distinct k bit prime divisors P and Q. Assume now that a coalition of players were able to produce a splitting of A into factors N_1, N_2 s.t. $P \mid N_1$ and $Q \mid N_2$. The probability that two (and thus polynomially many) randomly chosen numbers have a common large prime factor is negligible. Thus $\gcd(N_1, a_1)$ and $\gcd(N_2, a_1)$ yield a splitting of a large number a_1 into two factors, such that each of them contains a different large prime divisor of a_1.

Now a player could simulate the choices of the other random numbers $a_2, \ldots, a_r$ himself. He thereby had an algorithm that - with a non-negligible probability of success - splits a random number B, which has two large prime factors P and Q into factors which separate P and Q. This seems to be a completely infeasible task for all available factoring algorithms.

Theorem 1. *Let $\xi \in (\frac{1}{3}, \frac{5}{12})$. Then the number of integers $\leq x$ that have two distinct prime factors $\geq x^{\xi}$ is $x\big(\frac{1}{2}\ln^2 \frac{1}{2\xi} + O(\frac{1}{\ln x})\big)$.*

Proof. Let $A_t := \{y \leq x \mid t \mid x\}$ be the set of all multiples of t that are smaller or equal then x.

Let $G(x,\xi) = \bigcup_{p,q \geq x^{\xi}, p \neq q} \sharp A_{pq}$ be the number of all integers $\leq x$ that are divisible by 2 distinct primes $p, q \geq x^{\xi}$.

Furthermore let $\pi(x)$ denote the number of primes $\leq x$, ln the logarithm to the base e, and $\lfloor . \rfloor$ the greatest integer function.

Now let $\xi \in (\frac{1}{3}, \frac{5}{12})$. Then distinct sets A_{pq}, A_{rs}, for $p \neq q, r \neq s$ are disjoint as $A_{pq} \cap A_{rs} = A_{\mathrm{lcm}(p,q,r,s)}$ and $\mathrm{lcm}(p,q,r,s)$ has at least three prime divisors $> \sqrt[3]{x}$ in this case.

Hence

$$G(x,\xi) = \sum_{x^{\xi} \leq p < q \leq \sqrt{x}} \sharp A_{pq} = \frac{1}{2}\Big(\sum_{x^{\xi} \leq p,q \leq \sqrt{x}} \sharp A_{pq} - \sum_{x^{\xi} \leq p \leq \sqrt{x}} \sharp A_{p^2} \Big).$$

As $\sharp A_t = \lfloor \frac{x}{t} \rfloor$ we get

$$G(x,\xi) = \frac{1}{2}\Big(\sum_{x^{\xi} \leq p,q \leq \sqrt{x}} \lfloor \frac{x}{pq} \rfloor - \sum_{x^{\xi} \leq p \leq \sqrt{x}} \lfloor \frac{x}{p^2} \rfloor \Big)$$

$$= \frac{1}{2}\Big(\sum_{x^{\xi} \leq p,q \leq \sqrt{x}} \frac{x}{pq} + O(\pi^2(\sqrt{x})) - \sum_{x^{\xi} \leq p \leq \sqrt{x}} \frac{x}{p^2} + O(\pi(x)) \Big)$$

$$= \frac{1}{2}x\Big(\sum_{x^{\xi} \leq p,q \leq \sqrt{x}} \frac{1}{pq} - \sum_{x^{\xi} \leq p \leq \sqrt{x}} \frac{1}{p^2} + O(\frac{1}{\ln x}) \Big).$$

It is $\sum_{p \leq \sqrt{x}} \frac{1}{p^2} = O(1)$, as the series converges.

To estimate the other sum we use the well known number theoretic fact that there is a constant B s.t. $\sum_{p \leq y} \frac{1}{p} = \ln\ln y + B + O(\frac{1}{lnx})$ (cf. e.g. [5]). From this identity one obtains that $\sum_{y^{\xi} \leq p \leq \sqrt{y}} \frac{1}{p} = \ln\ln y^{\frac{1}{2}} - \ln\ln y^{\xi} + O(\frac{1}{lnx}) = \ln \frac{1}{2\xi} + O(\frac{1}{lnx})$.

Now

$$\sum_{x^{\xi} \leq p,q \leq \sqrt{x}} \frac{1}{pq} = \Big(\sum_{x^{\xi} \leq p \leq \sqrt{x}} \frac{1}{p} \Big)^2 = \Big(\ln \frac{1}{2\xi} + O(\frac{1}{lnx})\Big)^2 = \ln^2 \frac{1}{2\xi} + O(\frac{1}{lnx}).$$

All together we obtain

$$G(x,\xi) = \frac{1}{2}x\big(\ln^2 \frac{1}{2\xi} + O(\frac{1}{lnx}) \big)$$

$$= x\big(\frac{1}{2}\ln^2 \frac{1}{2\xi} + O(\frac{1}{\ln x})\big),$$

and the claim is proved.

Now we can describe an algorithm CONSTRUCT RSA-UFO that on input $(k, \frac{1}{2^t})$ outputs a $(k, \frac{1}{2^t})$-RSA-UFO modulus of bit length $O(kl)$. First a ξ that is slightly larger then $\frac{1}{3}$ is picked. Let k be sufficiently large. According to Theorem 1 there is a constant p depending only on the previous choice of ξ such that a randomly chosen number of bit length $\leq (\lfloor \frac{1}{\xi} \rfloor + 1)k$ is a k-RSA-UFO with probability $\geq p$. Thus one chooses a list $\mathcal{M}$ of $(\lfloor \frac{1}{p} \rfloor + 1)l$ random numbers of bit length $(\lfloor \frac{1}{\xi} \rfloor + 1)k$. This can be done by choosing each bit of their bit representation according to a public random source. Then the probability that none of the elements in $\mathcal{M}$ is a k-RSA-UFO is $\leq \frac{1}{2^t}$. Hence $N = \prod_{m \in \mathcal{M}} m$ is a $(k, \frac{1}{2^t})$ RSA-UFO modulus of bit length $O(kl)$ and the algorithm CONSTRUCT RSA-UFO outputs N.

A more careful determination of the constants will be contained in the full version of the paper.

Certainly there are some improvements to this general algorithm possible in order to obtain RSA-UFOs of smaller bit length. One may test the sampled numbers in $\mathcal{M}$ for primality and throw those found prime away. Furthermore one can divide out small prime factors from the elements in $\mathcal{M}$ to decrease the bit length of the output.

We think it's a theoretically interesting question if there is an algorithm that produces $(k, \frac{1}{2^t})$ RSA-UFO moduli of bit length smaller then $O(kl)$.

For practical applications it would be interesting to find RSA-UFOs of bit length as small as possible for parameters that give adequate security for practical applications. For example it would be interesting to have RSA-UFOs offering security comparable to usual 1024 bit RSA moduli. We ask: what is the smallest bit length of $(512, \frac{1}{2^{80}})$-RSA-UFOs? Our basic algorithm sketched above will yield an output of bit length (much) greater then 40.000 bits for these parameters, and may thus be too large for practical applications.

2.1 On the generation of public random strings

Our construction depends upon the availability of publicly verifiable random strings.

We describe an example to give an idea how such strings may be found in practice. Assume in November U.S. Congress publicly decides that such a string should be generated. Now Congress may fix a future date, e.g. noon of December 15 and decide on using the stock quotations at NYSE and Nasdaq at this future time for the generation of a random string. Also the procedure how to generate from the stock market data the random string is fixed by Congress in November. At noon of Dec. 15 a snapshot of the stock market is taken and using a (predetermined) hash function hashed to a shorter string. (If more random bits are needed one may also repeat this procedure on several (predetermined) dates and concatenate the obtained strings.)

The U.S. Congress decision is public and everybody knows when this snapshot will be taken. The stock market seems to be quite unpredictable and in particular it does not seem possible that any reasonable coalition of players is

able to manipulate the stock market in such a way that the output of the hash functions can be substantially influenced (e.g., forced to belong to a small set of values). The hash function should further extract the (assumed) randomness in the stock market data and produce bits that are close to uniform. Important for us is that later any party can verify that the procedure to construct the random string was correctly executed.

This is certainly only a heuristic argument, but it makes plausible that practically useful ways to generate a certified public random string seem to very well exist.

3 Accumulators From Generalized RSA Moduli

In the last section we described an algorithm that outputs RSA-UFOs. In this section we construct from this provably secure accumulators using the techniques developed in [1].

Intuitively an accumulator is secure if given a list $\mathcal{L}$ that hashed to an accumulated value z it is infeasible to produce a membership proof for an element x not belonging to $\mathcal{L}$. Assume we are given a (regular) RSA modulus N and a basis $x \in (\mathbb{Z}/N\mathbb{Z})^\times$ for the exponentiation. The original protocol to authenticate elements in a list $\mathcal{L} = \{y_1, \ldots, y_m\}$ via accumulators is not secure in the sense described above. Recall that to authenticate a value y Alice needs to present a y'th root of z to Victor. If Alice knows $\mathcal{L}$ she can not only authenticate the elements $y_1, y_2, \ldots, y_m$, which do belong to $\mathcal{L}$, but also, e.g., the element $y := y_1 \cdot y_2$, which may not belong to $\mathcal{L}$. A y'th root of z is given by

$$x^{\prod_{y \in \mathcal{L} \setminus \{y_1, y_2\}} y}.$$

More general: for every subset $S \subset \mathcal{L}$ the value $y_S := \prod_{y \in S} y$ can be authenticated by the value

$$x^{\prod_{y \in \mathcal{L} \setminus S} y}.$$

Although the list $\mathcal{L}$ contains only m items thus later potentially 2^m items can be authenticated later. This is clearly unsatisfactory.

Barić and Pfitzmann solve this (and more) problems by restricting the inputs to the accumulator function to prime numbers $e < N$. When Alice authenticates a value y she has as before to present a y'th root a of z. But in the verification step Victor not only checks that $a^y = z$ but also that $y < N$ and that y is prime.

From now on we will restrict ourselves to accumulators where the inputs are restricted to prime numbers and Victor checks primality during verification.

Barić and Pfitzmann introduce an even stronger notion of security for accumulators by requiring that it should be unfeasible to construct *any* list $\mathcal{L}$ such that a membership proof for an element not in $\mathcal{L}$ can be forged (thus here an adversary is allowed to choose the list $\mathcal{L}$ himself). In [1] an accumulator that achieves this requirement is called collision free.

This strong notion of security is appropriate for the applications sketched before. In the time stamping application it should not be possible to produce

time stamps on documents that have not been submitted at the time when their accumulated hash was computed, no matter how the submitted documents were chosen. In the electronic cash application users submit (hash values of) coins to the bank which inserts them as leaves into a hash tree. To assure that users are not able to forge money they should not be able to submit any set of coins to the bank, such that they can later prove that a new, forged coin belongs to the tree. To substitute trees by accumulators it is necessary to preserve this property also for accumulator based schemes.

Barić's and Pfitzmann's definition of collision freeness considers only adversaries, that do not know the factorization of N. Their accumulator is not collision free for parties that do know the trapdoor P and Q. We consider here a stronger definition of security by requiring that *no coalition* of polynomial time players should be able to forge membership proofs. Thus also bank insiders can not forge electronic coins and the time stamping agency can not forge time stamps. We call accumulators achieving this stronger security property *universally collision free*.

Definition 1. *A family of accumulators $(x, N, \mathcal{P})$ is called universally collision free if for all $m \geq 1$, it is infeasible for any coalition of probabilistic time bounded players to find a list $\mathcal{L} = \{y_1, \ldots y_m\} \subset \mathcal{P}$, a $y \in \mathcal{P}, y \notin \mathcal{L}$, and an $a \in (\mathbb{Z}/N\mathbb{Z})^\times$ such that a authenticates y, i.e. $a^y \equiv x^{y_1 \cdots y_m} \bmod N$, with non-negligible probability of success.*

In [1] a strong RSA assumption is formulated under which collision freeness of the accumulator against adversaries not knowing the trapdoor P, Q is proved. We formulate here a strong RSA-UFO assumption under which we then prove the security of the accumulator against any coalition of polynomial time players.

Strong RSA Problem: Given $x \in (\mathbb{Z}/N\mathbb{Z})^\times$ find a prime $e < N$ and an element s s.t. $s^e = x$.

Strong RSA-UFO Assumption: Let CONSTRUCT RSA-UFO be the probabilistic algorithm described in Section 2 that outputs a composite number N. Then for all probabilistic polynomial time algorithms $\mathcal{A}$, all polynomials P and all sufficiently large k:

$$Pr[a^e = x \bmod N \wedge e \text{ prime } \wedge \log e < 2k :$$

$$N = \text{CONSTRUCT RSA-UFO}(k, \frac{1}{2^k}); x \in_R \mathbb{Z}/N\mathbb{Z}; (a, e) \leftarrow \mathcal{A}(N, z)] < \frac{1}{P(k)}.$$

Unlike in the usual RSA assumption an adversary is also allowed to pick the exponent e.

We suggest here an RSA type accumulator, as described before, where the modulus N is an RSA-UFO, i.e. an output of the algorithm CONSTRUCT RSA-UFO with security parameters k and $\frac{1}{2^k}$, x is a random number modulo $\tilde{N}$ that has also been constructed according to a public random source, and the inputs to the accumulator are restricted to the set $\mathcal{P}$ of primes e with $\log e < 2k$.

Theorem 2. *Under the strong RSA-UFO assumption this accumulator is universally collision free.*

Proof. The proof proceeds as in [10, 2, 1]. Assume that given a RSA-UFO N and x an adversary can find a collision of the accumulator with the parameters (N, x), i.e. he can find a list of primes $\mathcal{L} = \{y_1, \ldots, y_m\} \subset \mathcal{P}$, a prime $y \notin \mathcal{L}, y \in \mathcal{P}$ and an element a s.t. $a^y = x^{y_1 \cdots y_m}$. As y is prime there are integers u, v s.t. $u(y_1 \cdots y_m) + vy = 1$, thus $(a^u x^v)^y = x$. Thus the adversary can extract an e'th root of x, for the prime $e = y$, breaking the strong RSA-UFO assumption.

3.1 On the strong RSA-UFO assumption

The strong RSA-UFO assumption is obtained from the strong RSA assumption formulated in [1] by considering RSA-UFOs $\tilde{N}$ instead of "usual" RSA moduli N. Note that variants of the (regular) strong RSA assumption have recently also been used in [4] and [3]. The strong RSA assumption has not been tested for too long although in [1] several arguments that could support the soundness of this assumption have been presented. We want to give here at least a relative argument: if one believes the strong RSA assumption it is also reasonable to believe the strong RSA-UFO assumption.

RSA moduli $N = PQ$ where P and Q are primes of the same bit length k are commonly used in cryptographic applications as they constitute the hardest known instances for factoring algorithms, where the running time is measured in the bit length of the input. We argue now that it is reasonable to believe that computing e'th roots of (random) elements $x \in (\mathbb{Z}/\tilde{N}\mathbb{Z})^\times$ for a $(k, \frac{1}{2^k})$- RSA-UFO $\tilde{N}$, where $\tilde{N}$ has been obtained as an output of CONSTRUCT RSA-UFO and $\log e < 2k$ is not easier then extracting e'th roots in $\mathbb{Z}/N\mathbb{Z}$ for usual RSA moduli $N = PQ$ for k-bit primes P and Q and $e < N$.

Assume an adversary were able to extract an e'th root of a random element x modulo $\tilde{N}$, where $\tilde{N}$ is an output of the algorithm CONSTRUCT RSA-UFO. Recall that $\tilde{N}$ was obtained as a product of randomly chosen large numbers $a_1, \ldots, a_r$. W.v.h.p. there is now one factor, say a_1, that has two distinct prime divisors P and Q of length at least k. Thus if an adversary can compute an e'th root y of $x \bmod \tilde{N}$ then, given a_1, he can in particular compute an e'th root $y \bmod a_1$ of $x \bmod a_1$. Now if x is a random number modulo $\tilde{N}$ then $x \bmod a_1$ is random number modulo a_1.

Thus even if an oracle would give an adversary a complete factorization of $\tilde{N}$ for free except the factorization of the divisor $P \cdot Q$ of a_1, the adversary would still be left with the problem to compute an e'th root of $x \bmod P \cdot Q$, where P and Q are primes of bit length at least k and $e < PQ$. This problem doesn't look any easier then computing an e'th root modulo a "true" RSA modulus which is the product of two primes of length exactly k.

3.2 Universally collision free accumulators under the random oracle assumption

Motivated by the fact that the strong RSA assumption still needs to be tested in [1], assuming the random oracle hypothesis, also a collision free accumulator under the *normal* RSA assumption is constructed. By replacing RSA moduli by RSA-UFOs the same construction (with the same proofs) yields easily an accumulator that can be proved to be universally collision free under the normal RSA assumption formulated for RSA-UFO moduli and the random oracle hypothesis. As this is completely analogous to the construction in [1] we omit here further details.

We further refer the reader directly to [1] for a discussion of a conversion algorithm that allows to convert (many) inputs into prime numbers, i.e. suitable inputs for the accumulator.

4 Conclusion

We showed how to make a provably secure accumulator without a (known) trapdoor. Our solution is based on an algorithm that outputs integers with large prime divisors such that "nobody knows a complete factorization". We think it is a theoretically and practically interesting question if there is an algorithm that produces such integers of a smaller bit length then the algorithm described in this paper.

5 Acknowledgments

It's a pleasure to thank Amnon Ta-Shma for many helpful and stimulating discussions that lead to this paper. I would further like to thank Stuart Haber and Moti Yung for interesting conversations on several aspects of this work.

References

1. Baric and Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT*, volume 1233, 1997.
2. Josh Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures (extended abstract). In Tor Helleseth, editor, *Advances in Cryptology—EUROCRYPT 93*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285. Springer-Verlag, 1994, 23–27 May 1993.
3. J. Camenisch and M. Michels. A group signature scheme with improved efficiency. *Lecture Notes in Computer Science*, 1514:160–??, 1998.
4. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski Jr., editor, *Advances in Cryptology—CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer-Verlag, 17–21 August 1997.

5. G. Hardy and E. Wright. An introduction to the theory of numbers. Oxford University Press, 5th edition, 1985.
6. M. Jakobsson and M. Yung. Revokable and versatile electronic mony. In Clifford Neuman, editor, *3rd ACM Conference on Computer and Communications Security*, pages 76–87, New Delhi, India, March 1996. ACM Press.
7. R. Merkle. Protocols for public key cryptosystems. In IEEE, editor, *Proceedings of the 1980 Symposium on Security and Privacy, April 14–16, 1980 Oakland, California*, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1980. IEEE Computer Society Press.
8. K. Nyberg. Fast accumulated hashing. In Dieter Grollman, editor, *Fast Software Encryption: Third International Workshop*, volume 1039 of *Lecture Notes in Computer Science*, pages 83–87, Cambridge, UK, 21–23 February 1996. Springer-Verlag.
9. T. Sander and A. Ta-Shma. Auditable, anonymous electronic cash. In *To appear in the proceedings of Crypto '99. Forthcoming volume in Lecture Notes in Computer Science*, 1999.
10. Adi Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Transactions on Computer Systems*, 1(1):38–44, February 1983.
11. S. von Solms and D. Naccache. On blind signatures and perfect crimes. *Computers and Security*, 11(6):581–583, October 1992.