Blind, Auditable Membership Proofs

Tomas Sander[1] *, Amnon Ta-Shma[2], and Moti Yung[3]

[1] InterTrust STAR Lab, Santa Clara, CA, USA
sander@intertrust.com
[2] International Computer Science Institute, Berkeley, CA, USA
amnon@icsi.berkeley.edu
[3] CertCo Inc., New York, NY, USA
moti@cs.columbia.edu

Abstract. Auditability is an important property in financial systems and architectures. Here we define the primitive of "blind auditable membership proof" (BAMP) which combines public auditability with privacy (i.e. user anonymity). In particular, one can use it as an auditable alternative to a "blind signature" component in unconditionally anonymous payment systems and in other systems requiring anonymity. We show that BAMP can be implemented quite efficiently (namely, without resorting to general zero-knowledge proofs of NP statements, which, in general, merely indicates plausibility).
We then build an anonymous off-line payment system based on the implementation of BAMP. The system has the property that its security against counterfeiting relies on the integrity of a public (auditable) database and not on the secrecy of privately held keys. The system strongly defends against blackmailing and bank robbery attacks, in the same way the system in [21] does. However, the current system is a significant step towards practicality since, unlike the previous system, first, it does not use general protocols for zero knowledge proofs for NP, and second, the cost of the payment protocol is independent of the number of total coins withdrawn.

1 Introduction

David Chaum [9] introduced the primitive of "blind signatures" in 1982 and showed how to build an anonymous electronic cash system based on this primitive. This primitive then served as the basic tool in implementing off-line payment systems which offer unconditional payer anonymity.

Although blind signatures are very elegant, appealing and relatively efficient, several drawbacks have been discovered over time due to new attack models. Van Solms and Naccache [24] discovered in 1992 a first serious attack on blind signature based payment systems. Their attack shows that a blackmailer can obtain anonymous electronic coins via anonymous communication channels. The withdrawn coin can not be distinguished from legitimately withdrawn electronic coins

* Work on this paper was done while author was at ICSI, Berkeley.

Y. Frankel (Ed.): FC 2000, LNCS 1962, pp. 53–71, 2001.

- thereby allowing for a "perfect crime". Various schemes that allow traceability (revocation of anonymity) have been designed to cope with this, and it was shown by Frankel, Tsiounis and Yung that revocable anonymity implies that anonymity cannot be unconditionally secure [14]. In 1996 Jakobsson and Yung [18] studied another potentially serious attack on anonymous payment systems, the bank robbery attack, in which the secret key of the bank is compromised and large sums of perfectly counterfeited electronic coins are injected into the system (this compromise can model the quite prevalent "internal attack") .

Both attacks above directly exploit features of the blind signature primitive. In fact, an anonymous payment system in which the validity of coins is determined by verifying signatures, is vulnerable to the blackmailing attack as blackmailers can force the bank into an unconditionally blind withdrawal protocol (if not efficiently then using secure computation as was pointed out in [18]). Furthermore, in a blind signature based payment system there is a highly sensitive secret key which is used by the bank to sign electronic coins and thus any such system is also potentially vulnerable to the bank robbery attack.

In [21] Sander and Ta-Shma suggested a different approach to off-line payment system where the security of the system is based on public auditing. In a nutshell the approach is based on the use of "membership proofs" instead of signature techniques: during payment the payer sends the coin to the merchant together with a proof that the coin belongs to a public list of "valid coins". This list is managed by the bank and can also be publicly audited (typical audit is done by a number of public independent entities). Furthermore, they show how such a proof can be given in a "blinded" way, i.e., such that transcripts of withdrawal and payment are statistically independent and the resulting payment system offers users unconditional anonymity. The system in the approach of [21] is no longer susceptible to the bank robbery attack as its security against counterfeiting relies on the integrity of a public database and not on the secrecy of secret keys. This notion of audit is quite close to various real life Auditing scenarios which are performed by independent entities on an available data. Furthermore, the system also defends to a high degree against blackmailing since the bank can always invalidate illegitimately withdrawn coins.

1.1 Our results

In this paper we formalize the central concept behind the approach of [21] and isolate it as a primitive which we call "blind auditable membership proofs" (BAMP). The identification of the primitive is important since it can be employed as a general technique whenever anonymity (i.e., individual privacy or traceability-freeness) and auditing need to be combined.

The system implementation of the payment approach suggested in [21] was based on the use of Merkle hash trees: during withdrawal randomized hash values of the serial numbers of the coins are inserted as leaves in the tree. The leaves of the tree correspond to valid coins. During payment a serial number is revealed and the payer proves that this serial number appears in a leaf of the "tree of valid coins". Blinding of this membership proof and thereby unconditional payer

anonymity was achieved by using general protocols for zero knowledge proofs (arguments) for NP. Here we improve on this implementation in two ways. First, we get rid of the tree structure that causes the complexity of the payment step to depend poly-logarithmically on the total number of withdrawn coins. Secondly (and perhaps more importantly) we use efficient zero knowledge proof techniques and have no longer to recur to zero knowledge proof techniques for NP. We achieve this by implementing "blind membership proofs" in an algebraic setting. This is a crucial step, advancing the state of the art from "plausibility argument" towards efficiency (i.e. the possibility of an implementable construction). This basic argument distinguishing plausibility results from efficient ones was put forth in [15, 22]. (On the other hand, we do not want the reader to assume that we claim that our work is the last word on efficiency and practicality of the suggested notion and approach.)

1.2 Overview of our construction

The first basic ingredient of our solution is the one way accumulator construction of Benaloh and deMare [3] that allows to prove membership in a list $\mathcal{L}$ efficiently. The basic idea is as follows. Let N be an RSA modulus and $x \in Z_N$ be a random element. Let $\mathcal{L} = \{a_1, \ldots a_m\}$. The accumulated hash value z of $\mathcal{L}$ is defined to be the value $z = x^{a_1 \cdot a_2 \cdots a_m} \bmod N$. Assume now that Victor has obtained z over an authenticated channel. To prove membership of an element a in $\mathcal{L}$, Alice presents to Victor an a'th root w of z. Victor accepts if $w^a = z$ (for brevity we omit the modulo notation hereafter).

Barić and Pfitzmann (following Shamir [23]) introduced in [1] the strong RSA assumption. They showed that under this assumption this accumulator protocol can be proved secure, if one restricts the elements of $\mathcal{L}$ to prime numbers smaller than the modulus N. Thus during verification Victor needs not only to check that $w^a = z$ but also that a is prime. Variants of the strong RSA assumption have recently been used in several schemes [16, 6, 17, 11].

In order to be able to authenticate arbitrary numbers (and not only prime numbers) via this accumulator protocol we need a way to "convert" arbitrary numbers into prime numbers that are suitable for the accumulator. This conversion was studied by Halevi et al. [17] as a subroutine for a different construction, and we adopt their solution. During system setup a hash function h is randomly chosen from a 2-universal family of hash functions. For any element a, it is possible to efficiently find a large prime number p, such that $h(p) = a$. Instead of feeding a directly into the accumulator, the prime p is fed into the accumulator. Alice authenticates a by presenting (a, p, w) and Victor accepts if $h(p) = a$ and $w^p = z$. Note that in our adaptation of this protocol Victor does not need to test the primality of p. We further give an efficient algorithm which computes the modular roots w_i, s.t. $w_i^{p_i} = z$ for a large list $\mathcal{L} = \{p_1, \ldots p_m\}$.

The sketched accumulator protocol, so far, is certainly not blind, since a verifier sees the element a. To turn it into a blind protocol, we have Alice commit herself to the values a, w via unconditionally hiding and computationally binding

commitments C_a and C_w, respectively. Alice then proves that for these committed values the relation $w^a = z$ holds. The protocols of Camenisch and Michels [7] allow us to implement the modular exponentiations on secret, committed values (quite) efficiently. This yields a basic efficient blind proof of membership in $\mathcal{L}$. Alice can prove efficiently and independently of the number of elements in $\mathcal{L}$ that she knows an element in $\mathcal{L}$ without revealing the element. We note that the fact that Victor does not need to check that p is prime is important and saves us the need to implement an efficient blind proof that a committed number is prime.

To make use of this construction for an anonymous payment scheme the value a should contain further information such as the serial number of the coin and an appropriate encoding of the identity of the user to detect double spenders in an offline system. More precisely: we let a be an element of a large multiplicative subgroup G_q of prime order of a finite field. Let $g_1, \ldots, g_k$ be randomly chosen generators for G_q and $(u_1, \ldots, u_k)$ be a representation of a w.r.t. this base, i.e. $a = g_1^{u_1} \cdots g_k^{u_k}$. During withdrawal, information like the serial number is embedded in this representation and the appropriate information is revealed during payment.

This yields an efficient, anonymous payment system. It is secure against the blackmailing attack in the sense described in [21]. Similarly, the security of this payment system against counterfeiting and the bank robbery attack relies upon the ability of the bank to distribute accumulated hash values securely. With respect to the bank robbery attack we pay a certain price for our efficient algebraic construction. During the system set up the RSA modulus N needs to be constructed. In the currently known algorithms to construct N, the parties constructing the RSA modulus $N = PQ$ can also find the prime factors P, Q of N. Knowledge of this "trapdoor" of the accumulator translates directly into the ability of giving false membership proofs (and thereby to "forge") coins. Thus the factors P and Q should be chosen in an isolated process (trusted dealer) and be destroyed after system setup as in [10]. Alternatively and in some respects more securely, a distributed generation of the RSA modulus is possible (see [4, 13]).

Unlike in blind signature based payment systems where the sensitive secret signature key of the bank is needed in each withdrawal session, no secret information is needed during the operation of the payment system described in this paper. Thus if the trapdoor information is reliably destroyed during system set up (by the centralized/ distributed holders of the factors) security against bank robbery is, in fact, achieved. We also note that an RSA type accumulator construction without trapdoor was given by Sander in [20]. In principal we could use this trapdoor free accumulator construction and achieve by this a strong defense against the bank robbery attack, however this construction is less efficient (due to modulus expansion).

1.3 Organization of the paper

In Section 2 we define the primitive of a blind auditable membership proof. In Section 3 we describe some of the basic tools and assumptions needed for our construction. In Section 4 we describe a provably secure and efficient accumulator. In Section 5 we describe a protocol for a blind auditable membership proof. Finally, in Section 6 we describe an efficient payment system based on the ideas of the earlier sections.

2 Blind, Auditable Membership Proofs

In this section we first define "auditable membership proofs" and then augment it to define "blind auditable membership proofs".

Definition 1. (Auditable membership proof) *Let A be a set of elements. Let $\mathcal{L}$ be the set of all ordered lists over A. An auditable membership proof for A, is a triple (F, G, V) s.t. $F : \mathcal{L} \to Z$, $G : \mathcal{L} \times A \to W$ and $V : A \times W \times Z \to \{\text{True}, \text{False}\}$ s.t.*

- *(completeness) $\forall L \in \mathcal{L}, \forall a \in L \quad V(a, G(L, a), F(L)) = \text{True}$.*
- *(soundness) It is infeasible for* any *coalition of polynomial time players to find a list $L \in \mathcal{L}$, an element $a \notin L$ and $w \in W$ s.t. $V(a, w, F(L)) = \text{True}$.*

We say the membership proof is efficient *if F, G and V are polynomial time algorithms.*

Let us now give two concrete constructions which comply with the above definition:

Example 1. (Merkle hash trees.) $F(L)$ is the value z at the root of a hash tree that contains the elements of L at its leaves. $G(L, a)$ is the hash path from a to the root. $V(a, w, z)$ is the path verification algorithm which is True iff w is a hash path from a to z.

Example 2. (Accumulators.) The basic idea of accumulator based auditable membership proofs is as follows: Let $N = PQ$ be a RSA modulus and let $a \in Z_N^*$. We want to prove membership to $L = \{a_1, \ldots a_k\} \subset Z_N^* \setminus \{1\}$. The "accumulated hash value" $z = F(L)$ of the list L is defined to be the value $z = x^{a_1 \cdot a_2 \cdots a_k} \bmod N$. $G(L, a)$ is an a'th root w of z. $G(L, a)$ can be computed by raising x to the power $b = \prod_{y \in L \setminus \{a\}} y$. $V(a, w, F(L)) = \text{True}$ if and only if $w^a = F(L)$. In Section 4 we will discuss how this basic scheme can be made provably secure.

We now define a *blind*, auditable membership proof. This time we define it as a multi-player protocol. Let us have k players $P_1, \ldots, P_k$, one central player B and a verifier C. Each player P_i sends an element $a_i \in A$ to B, where A is the set from which elements are withdrawn, A is known and finite. As before we have the F, G, V functions: $F : \mathcal{L} \to Z$ is the hash function B uses to hash the

list. $G : \mathcal{L} \times A \rightarrow W$ is the function B uses to take a list $L \in \mathcal{L}$ and an element $a \in A$ and return the "witness" w that $a \in L$. $V : A \times W \times Z \rightarrow \{\text{True}, \text{False}\}$ is the predicate used to verify that $a \in A$ indeed belongs to the list that was hashed to $z \in Z$.

Now, we would like Alice to prove she holds some $a \in L$ without actually revealing what a is. This is, often, not too useful, because the list is public and anyone can know an element a from the list. Thus, we want in addition that Alice proves "ownership" of the element a, or more generally, that some predicate Q holds for the input a. This predicate can be, e.g., that Alice knows a preimage of a under a certain hash function, or that a is a large or small number, or, in general, any property of a that can be evaluated by a small arithmetic circuit.

Definition 2. (Blind, auditable membership proof) *A blind, auditable membership proof is a protocol between k players $P_1, \ldots, P_k$, one central player B and a verifier C, where k is at most polynomial in the protocol's security parameter.*

Setup: *The protocol begins with each P_i having a private input $s_i \in S$ and a public value $a_i \in A$ (the sizes of elements are polynomial in the security parameter).*

Building the list: *Player P_i communicates a_i to B. After all players communicated their values, B computes $z = F(a_1, \ldots, a_k) \in Z$ and $w_1, \ldots, w_k \in W$ where $w_i = G(a_i, \{a_1, \ldots, a_k\})$. B makes z public, and sends w_i to player i. The list may or may not be public.*

Proof: *P_i sends C a value t_i. P_i and C then execute a (possibly interactive) protocol and C either accepts or rejects.*

It should hold that:

Completeness: *If a player P knows a, s and t s.t $a \in L$, $Q(s,t,a) = \text{True}$, then after execution of the protocol C accepts.*

Soundness: *For any coalition of polynomial time players, for all values $a_1, \ldots, a_l$ they choose to submit to the list the following holds: there is a knowledge extractor, s.t. if C accepts, the knowledge extractor can find in expected polynomial time values a, s s.t. $a \in L$ and $Q(s,t,a) = \text{True}$ given the data the coalition knows. (The acceptance probability and extraction probability may differ by a negligible soundness error probability).*

Blindness: *Let $\mathcal{T}$ be the history of protocol execution transcripts. Suppose a honest P_i executes a "proof" protocol with C for proving knowledge of $a \in L$,s and t. Let us denote by $\mathcal{D}_{\mathcal{T},i,a,s,t}$ the distribution of the transcript of the protocol.*

We say the protocol is statistically ϵ–blind, if for any t there is one fixed distribution $\mathcal{D}_t$ s.t. for any history, any honest player P_i, any $a \in L$ and s, $|\mathcal{D}_{\mathcal{T},i,a,s,t} - \mathcal{D}_t| \leq \epsilon$. (Namely, the transcript distribution does not depend on the history, the data or the player).

We note that a dynamic version of the above definition is possible where z is constructed incrementally (in discrete time units).

3 Tools

3.1 Universal-2 hash functions

Definition 3. *[8] A family $H = \{h : A \rightarrow B\}$ is 2-universal if for every $a_1, a_2 \in A, a_1 \neq a_2, b_1, b_2 \in B$*

$$\Pr_{h \in H}(h(a_1) = b_1 \wedge h(a_2) = b_2) = (\frac{1}{|B|})^2$$

Example 3. We take a set $H = \{h : \{0,1\}^k \rightarrow \{0,1\}^m\}$. Each function $h \in H$ is indexed by a $m \times k$ matrix A and a vector $b \in \{0,1\}^m$. For $h = h_{A,b}$ we define $h(x) = Ax + b$. It is well known that H is a universal-2 family.

H has the additional property that given $h = h_{A,b} \in H$ and $z \in \{0,1\}^m$ it is easy to sample a random element from $h^{-1}(z)$, which is just the problem of picking a random solution to the linear system $Ax + b = z$.

If $A = \{a_1, \ldots, a_k\}$ and we define a random variable $X_i = h(a_i)$ that is obtained by picking h uniformly from H and computing $h(a_i)$, then $X_1, \ldots, X_k$ are pair-wise independent, i.e., for every $1 \leq i < j \leq k$, X_i and X_j are independent.

3.2 The strong RSA assumption

Baric and Pfitzmann [1] define the following problem asserting that RSA is simultaneously hard to invert on any potential exponent:

Strong RSA Problem: Given $x \in Z_N^*$ find an $e, 2 \leq e < N$ and an element s s.t. $s^e = x$.

Strong RSA Assumption: $\mathcal{B}$ is a probabilistic algorithm that on input 1^k outputs a RSA modulus N of size k uniformly at random. For every probabilistic polynomial time algorithm $\mathcal{A}$, every polynomial P, for all sufficiently large k:

$$Pr[a^e = x \bmod N \wedge 2 \leq e < N :$$

$$N = \mathcal{B}(1^k); x \in_R Z_N; (a, e) \leftarrow \mathcal{A}(N, x)] < \frac{1}{P(k)}.$$

3.3 The group representation problem

Let G_q be a group of prime order q for which DLOG is hard, and let $g_1, \ldots, g_s$ be known, randomly chosen elements from G_q. We say $a \in G$ has a representation $(a_1, \ldots, a_s)$ with respect to the basis $(g_1, \ldots, g_s)$ if $a = g_1^{a_1} \cdot \ldots \cdot g_s^{a_s}$. In [5] an efficient protocol is described to prove knowledge of a representation of an element a w.r.t. a known basis $(g_1, \ldots, g_s)$. The protocol does not reveal any further information.

3.4 Proving modular relations in zero knowledge

In this paragraph we briefly describe efficient building blocks for our later constructions. We need protocols for arithmetic over secret committed values modulo various moduli $N_1, N_2, \ldots N_s$, and we use here special statistical zero knowledge arguments of knowledge from Camenisch and Michels [7]. These tools can be used to eliminate general ZK-proof techniques for NP when dealing with algebraic structures. See [7] for a more detailed description of these tools.

The setting. Let l be a positive integer. Let N be an integer s.t. $0 < N < 2^l$. We will assume further that the values a, b, c for which we want to prove modular relations modulo N fulfill the condition $-2^l < a, b, c < 2^l$. This range condition can be enforced by the protocol. Let Q be a prime s.t. $Q > 2^{2l+5}$. Let G be a group of order Q s.t. computing discrete log in G is hard (G can be chosen to be a subgroup of the multiplicative group of a large finite field F_P). Let g, h be two generators of G such that $\log_g h$ is not known.

Commitments. We commit to a value $a \in Z_Q$ by $C_a := g^a h^r$, where $r \in_R Z_Q$. This commitment scheme is unconditionally hiding and computationally binding (assuming DLOG is hard).

The building blocks. Quite efficient statistical zero knowledge arguments of knowledge for many modular relations (e.g., addition, multiplication, exponentiation) on the commitments are described in [7]. The techniques also allow one to prove the correctness of the disjunction of statements about discrete logs without revealing which of the statements is true. This allows to prove that a committed value v encodes a single boolean bit, by proving the statement "(C_v is a commitment of 0) $\vee$ (C_v is a commitment of 1)". Thus, one can commit to a value a in several different ways and prove that they encode the same value. E.g., one can commit to $a = a_1, \ldots, a_k$ bit by bit, later on commit to it as an integer, and then prove that the value $\Sigma_i a_i 2^i$ when computed from the committed values in C_{a_i} equals the value committed to by C_a. The relations that we need are:

1. (Linear relations): $a + b \equiv c \bmod N$, or more generally, $\Sigma p_i a_i = b$, where the p_i are public and the values N, a, b, c, a_i may be committed.
2. (Multiplication): $a * b \equiv c \bmod N$ where the values N, a, b, c may be committed.
3. (Exponentiation): $a^b \equiv c \bmod N$ where the values N, a, b, c may be committed.
4. (Equality): $a = b \bmod N$ where a, b, N may be committed.
5. (Non-Equality): $a \neq 0 \bmod N$ where a, N may be committed.
6. (Equivalence of commitments): A commitment to a binary string carries the same value as another commitment to a non-binary value.
7. (Opening a commitment): C_a is the commitment of a public value a.

3.5 Transferable ZK-proofs

Finally, following an idea of Fiat and Shamir [12], that was formalized using the random oracle assumption in [19, 2], we convert interactive, zero-knowledge proofs to non-interactive, transferable zero-knowledge proofs, by replacing the challenges with an output of the random oracle on the initial commitments.

4 A provably Secure Accumlator Construction

Let N be an RSA modulus. Let $H = \{h : \{0,1\}^k \to \{0,1\}^m\}$ be a universal-2 family of hash functions, where $\frac{N}{2} \leq 2^k < N$ and $k \geq 3m$. We further assume that given $h \in H$ and $a \in \{0,1\}^m$ it is easy to sample a random element from $h^{-1}(a)$. Our accumulator is based on [3, 1] and [17].

Submission: Player P_i sends $a_i \in \{0,1\}^m$ to the list manager B. B forms the list $\mathcal{L} = \{a_1, \ldots, a_l\}$.

Hashing: B chooses a random element x from Z_N. For each $i \in \{1, \ldots, l\}$ B randomly samples $h^{-1}(a_i)$ until B finds a large prime number $p_i \in [1..2^k]$, $p_i \geq \sqrt{N}$, s.t. $h(p_i) = a_i$.

B computes $z = x^{p_1 \cdot p_2 \cdots p_l} \bmod N$ and makes z and x public; z is the "hash" of the list $\mathcal{L}$. B computes $w_i = x^{\prod_{j \neq i} p_j}$ and sends the triple (a_i, w_i, p_i) to the i'th player.

Membership proof: C knows the values x and z but not necessarily the values of $\mathcal{L}$. A wants to prove to C that an element a belongs to the list $\mathcal{L}$. To do that A sends the triple (a, w, p) to C. C accepts iff the following three conditions hold:

1. $\sqrt{N} < p < N$,
2. $h(p) = a$, as binary vectors,
3. $w^p = z (\mathrm{mod} N)$.

We stress that C does not need to check that p is prime.

Theorem 1. *Under the strong RSA assumption the above protocol is a complete and sound membership proof protocol.*

Proof.

Completeness: We need to show that w.h.p. B will almost surely find a big prime p_i in $h^{-1}(a_i)$. This follows from the following lemma (due to [17]).

Lemma 1. *For any $A \subseteq \{0,1\}^k$, for all but a negligible $O(\frac{2^{2m}}{|A|})$ fraction of $h \in H$, for all $z \in \{0,1\}^m$, $\Pr_{x \in h^{-1}(z)}(x \in A) \geq \frac{\rho(A)}{2}$ where $\rho(A) = \frac{|A|}{2^k}$.*

For complete exposition we prove the lemma in the appendix. Now, since (by the prime number theorem the density of big primes in Z_{2^k} is at least $\Omega(\frac{1}{k})$, for almost all $h \in H$ (except for, may be, $2^{-\Omega(k)}$ fraction) the expected number of samples required to find a big prime $p_i \in h^{-1}(a_i)$ is $O(k)$ and sampling $O(k^2)$ times may miss a prime number with only a negligible $2^{-\Omega(k)}$ probability.

Soundness: We show that membership proofs can not be forged. I.e., x is first chosen at random from Z_N. Then, we let our adversary Eve choose the elements $a_1, \ldots, a_l$ that are submitted to B. B computes $p_1, \ldots, p_l$ and $z = x^{p_1 \cdots p_l}$. We want to show that Alice can not find elements a, p and w that prove membership for a not already in the list.

So suppose Eve can find values z, a, p, w and a_i, p_i, w_i $i = 1, \ldots, l$ s.t.

- p_i is a prime, $\sqrt{N} < p_i < N$, $h(p_i) = a_i$, and $w_i^{p_i} = z \mod N$.
- $\sqrt{N} < p < N$, $h(p) = a$ and $w^p = z \mod N$. We stress that here we do not require that p is a prime.

The values p_i must be prime and fulfill the range condition because by auditing the list manager this can be enforced.

Denote $d = \gcd(p, p_1 \ldots p_k)$. Thus, $gcd(\frac{p}{d}, \frac{p_1 \cdots p_k}{d}) = 1$. Define $e = \frac{p}{d}$. There are integers $u, v \in Z$ such that $ue + v\frac{(p_1 \cdots p_l)}{d} = 1$ holds over the integers, and moreover Eve can find them in polynomial time using the extended GCD algorithm.

Now set $s = w^v x^u$. Then

$$\begin{aligned} s^e &= w^{ve} x^{ue} = w^{\frac{vp}{d}} x^{ue} = z^{\frac{v}{d}} x^{ue} \\ &= x^{v\frac{(p_1 \cdots p_k)}{d} + ue} = x \end{aligned}$$

Thus Eve can find, in polynomial time, a value s which is an $e'th$ root of x. By the strong RSA assumption it must be that $e = 1$. Hence, $p = d$, i.e., $p_1 \cdots p_d | p$. However, $p < N$ and each $p_i > \sqrt{N}$, thus it must be that $p = p_i$ for some $i \in \{1, \ldots, l\}$. In particular, $a = h(p) = h(p_i) = a_i$, i.e., a is already in the list. This completes the proof that Eve can find membership proofs only for elements already in the list.

4.1 An algorithm for computing w_j

We conclude with an efficient algorithm for computing the witnesses w_j for $j = 1, \ldots, l$. The algorithm works even when $\phi(N)$ (and the factorization of N) is not known. The trivial algorithm requires $O(l^2)$ modular exponentiations, and we show how to employ divide and conquer to do this with only $O(l \log(l))$ modular exponentiations.

- Input: $\{p_1, \ldots, p_l\}$, N, $x \in Z_N$.
- Output: $w_1, \ldots, w_l$, $w_j = x^{\prod_{i \neq j} p_i}$.

Algorithm 2 *W.l.o.g. we assume l is a power of two. Given $\{p_1, \ldots, p_l\}$ we compute*

- $A = x^{p_1 \cdots p_{l/2}}$ *and*
- $B = x^{p_{l/2+1} \cdots p_l}$.

We then recursively solve the following two problems:

- *The input is the set $\{p_1, \ldots, p_{l/2}\}, N$ and B. This gives us all w_j for $j \in \{1, \ldots, l/2\}$.*
- *The input is the set $\{p_{l/2+1}, \ldots, p_l\}, N$ and A. This gives us all w_j for $j \in \{l/2 + 1, \ldots, l\}$.*

Altogether we get all w_j for $j \in \{1, \ldots, l\}$. If we denote the complexity (number of modular exponentiations) of the algorithm for l elements by $T(l)$ then $T(l) = 2T(l/2) + O(l)$, $T(1) = 1$. Thus, $T(l) = O(l \log(l))$.

5 A Protocol for Blind, Auditable Membership Proofs

We now give a protocol implementing the blind auditable membership proof (BAMP) with respect to a predicate Q (which is an algebraic expression over its parameters).

Protocol 1 *(Blind auditable membership proofs)*
System Setup: *During system setup the following parameters are chosen by a trusted dealer or a group:*

1. *an RSA modulus N with unknown factorization and a random element $x \in Z_N$,*
2. *a random element $h \in H$ from the 2-universal family of hash functions, $H = \{h : \{0,1\}^k \rightarrow \{0,1\}^m\}$. We require that $k \geq 3m$ and k is the largest integer such that $2^k < N$. We use the specific construction of Example 3, i.e., $h(x) = Ax + b$.*

Submission: *Player P_i generates $a_i \in \{0,1\}^m$ and sends a_i to the list manager B. B forms the list $\mathcal{L} = \{a_1, \ldots, a_l\}$.*
Hashing: *B chooses prime numbers $p_i \in [1..2^k]$ such that $p_i \geq 2\sqrt{N}$ and $h(p_i) = a_i$. B computes $z = x^{p_1 \cdot p_2 \cdots p_l} \bmod N$ and makes z and x public; z is called the "hash" of the list $\mathcal{L}$. B computes $w_i = x^{\prod_{j \neq i} p_j}$ and sends the triple (a_i, w_i, p_i) to the i'th player. The player P_i checks that $A(p_i) + b = a_i$ as binary vectors and that $w_i^{p_i} = z \bmod N$.*
Membership proof: *C knows the value z. A sends the value t to C. A gives a zero knowledge proof of knowledge of elements a, w, p, s of the following statement:*

1. *p lies in the range $[2^{\frac{k}{2}+1}, 2^k]$, and,*
2. *$Ap_{bin} + b = a_{bin}$, where p_{bin}, a_{bin} are the binary expansions of p and a, and,*
3. *$w^p = z \bmod N$, and,*
4. *$Q(s,t,a) = \text{True}$.*

5.1 Efficient implementation

Let $p_{bin} = p_0 \ldots p_k$ be the binary expansion of p and $a_{bin} = a_0, \ldots, a_m$ be the binary expansion of a. A commits to p, a, w, s and commits bitwise to the binary expansion of p and a using the commitment protocol described in Section 3. We denote these commitments by $C(p), C(a), C(w), C(p_i), C(a_i)$. A then gives a statistical ZK proof of knowledge that:

- The values that are committed in $C(p_0), \ldots C(p_k)$ (resp. $C(a_0), \ldots C(a_m)$) are in fact the binary expansions of the values committed to in $C(p)$ (resp. $C(a)$) using the Equivalence of Commitments sub-protocol.
- The range condition on the value committed in $C(p)$ by checking that the $k/2 - 1$ most significant bits of p are not all zero, using the Non-Equality sub-protocol.

- The relation $Ap_{bin} + b = a_{bin}$ holds for the corresponding committed values using the Linear Relations sub-protocol.
- The relation $w^p = z (\mathrm{mod} N)$ holds for the committed values $C(w), C(p)$ using the exponentiation sub-protocol.
- The relation $Q(a, s, t)$ holds for the committed values and the public value t (we assume that Q can be described by a "simple" algebraic circuit).

Using a random oracle hash function, the proofs above (which are based on random bit challenges by the verifier) can be turned into non-interactive ones of the Fiat-Shamir type.

5.2 Proof of properties

Theorem 3. *Assuming the strong RSA assumption, Protocol 1 gives an efficient blind auditable membership proof, which can be made non-interactive under the random oracle assumption.*

Proof.
Completeness: Since the used basic protocols are complete the entire protocol is obviously complete.
Soundness: Suppose A can convince C to accept. By the definition of proofs of knowledge there is a polynomial time knowledge extractor such that given the data A holds outputs values a, p, s, w where the relations $w^p = z, Ap_{bin}+b = a_{bin}$, p lies in the range $[2^{\frac{k}{2}+1}, 2^k]$ and $Q(a, s, t) =$ True are all true. As $p \in [2^{\frac{k}{2}+1}, 2^k]$, $h(p) = a$ and $w^p = z$. Theorem 1 implies that a is one of the values in the list whose hash is z.
Blindness: The zero knowledge proofs reveal that A knows a, w, p, s and a public value t, where conditions (1-4) of the membership proof hold, and nothing more (formally, this means that there is a polynomial time simulator that can produce an almost identical distribution based on the known public values). Now, any honest player who makes C accept knows some a, w, p, s, t such that (1-4) hold. Thus, the actual information C gets is the value t.

Therefore, there is a simulator that given t generates a distribution $\mathcal{D}_t$ that is statistically close to the actual distribution of the transcript of the interaction between A and C (or the transcript of "interaction" with the random oracle).

6 An Efficient Off-Line Ecash System

Here we build an auditable, off-line payment system. It follows the ideas in [21], but its core data structure is based on accumulators rather than hash tress. In our system each user has one fixed identity (P_A, S_A) where S_A serves as a secret key that remains private even in the case of double spending, and one double-spending identity D_A that gets revealed in case of double spending. Even if a user double spends he can not be framed for double spending he has not done.

6.1 The protocol

Bank's setup:

- (Choosing a group G_q) The bank chooses large primes p, q, s.t. $p = cq + 1$ for some integer c (e.g., $c = 2$). G_q is the subgroup of order q of Z_p^*. The bank picks random elements $g_1, g_2, g_3, g_4, g_5 \in_R G_q$.
- (Choosing an accumulator) The bank chooses an RSA modulus $N = p_1 p_2$, where p_1, p_2 are two big primes, where $N \approx p^3$. The bank also chooses a random $x \in Z_N^*$.
- (Choosing a universal-2 hash function) $G_q \subseteq Z_p$ and therefore there is a natural embedding of elements of G_q (and Z_p) as a binary string in $\{0,1\}^m$ (so we pick m to be the smallest integer s.t. $2^m \geq p$). The bank chooses a universal-two family of hash-functions $H = \{h : \{0,1\}^k \rightarrow \{0,1\}^m\}$, with k the largest integer s.t. $2^k \leq N$, and picks $h \in H$ at random. Notice that $k \geq 3m$.
- (Choosing a hash function) The bank also uniformly selects a hash function $\mathcal{H}$ from a collection of collision intractable hash functions.

The bank makes $p, q, g_1, g_2, g_3, g_4, g_5, N, m, k, h, \mathcal{H}$ public. The bank should destroy p_1 and p_2. Note that the above system parameters can also be chosen by a trusted (and distributed) organization.

Account opening: Alice chooses $S_A \in_R Z_q$ and computes $P_A = g_1^{S_A} \in G_q$. Alice also chooses $D_A \in_R Z_q$. Alice identifies herself along with the numbers D_A and P_A, and proves to the bank that she knows a representation for P_A in the g_1 basis. The bank records Alice's identity together with (D_A, P_A).

Withdrawal: Alice identifies herself to the bank. Then she picks $u_1, u_2, serial \in_R Z_q$ and computes $T = g_3^{u_1} g_4^{u_2} g_5^{serial}$. Alice sends T to the Bank along with a proof of knowledge of a representation of T according to the basis (g_3, g_4, g_5) [5]. Both sides set $a = P_A \cdot g_2^{D_A} \cdot T$. In particular Alice knows a representation $(S_A, D_A, u_1, u_2, serial)$ of a according to the basis $(g_1, \ldots, g_5)$. Then the bank finds a large prime $p' \in h^{-1}(a)$, $p' \in \{0,1\}^m$, $2\sqrt{N} \leq p' < N$. By Lemma 1 the bank can efficiently find such a value p' after not too many samples. The bank records that user (P_A, D_A) obtained a coin a, and deducts the corresponding amount from her account.

When the time frame ends (say, every minute) the bank takes all the values $\{(a_i, p_i')\}$ received at that time frame and computes $z = x^{\prod_i p_i'} (\mathrm{mod} N)$. The bank also computes $w_j = x^{\prod_{i \neq j} p_i'} (\mathrm{mod} N)$, for all j using Algorithm 2. Then the bank sends (p_j', w_j, z) to player j. The player checks that

- $h(p_j') = a_j$ as binary vectors, where a_j is the value he submitted to the bank.
- p_j' is a prime with $2\sqrt{N} \leq p_j' < N$.
- $(w_j)^{p_j'} = z (\mathrm{mod} N)$.

The bank will also send updates to the user and we describe this next.

Updates: Every minute a new 'minute' list is formed. When two minute lists exist they are combined into an 'hour' list. When two hour lists exist they are combined into a 'day' list, and so forth. Each time two lists are combined the bank computes the hash z of the combined list, and new witnesses w_i, using Algorithm 2, and sends an 'update' message with (w_i, z) to each user P_i who has a coin in the combined list. We analyze the complexity of this soon. Thus, each user gets a minute/hour/day/month etc. update for his coin, when the time comes.

We say an accumulation (or a hash) z is *alive* if it is a hash of the current minute/hour/day etc. list. There are at most, say, 60 live accumulations. Each merchant can choose how often to be updated about the set of live accumulations. A merchant who chooses to be updated only once a day, can accept coins only from users who withdrew their coin at least a day ago. For more details see [21]. Unlike the system in [21] we do not use broadcast in our update system.

Now we analyze the complexity of computing the updates. Each time the bank combines two lists into a list of size l, the bank performs $O(l \log(l))$ modular exponentiations. We now group together all the operations needed to compute witnesses on the minute level, and we see that the minute level requires at most $O(c \log(c))$ modular exponentiations, where c is the number of coins. Similarly, any level (hour, minute, day etc.) requires at most $O(c \log(c))$ modular exponentiations. We see that altogether the system requires $O(c \log^2(c))$ modular exponentiations. That is, the bank has to execute $O(\log^2(c))$ modular exponentiations per withdrawn coin.

Payment: Alice first commits to:

- the value p' both as a binary vector and as an element of Z_N.
- the value w as an element of Z_N.
- the value a both as a binary vector and as an element of Z_p.
- the values S_A, D_A, u_1, u_2 as elements Z_q.

Alice then computes the challenge c as $c = \mathcal{H}(Merchant_{id}, time, commitments)$, and sends $c, serial, v \in Z_q$ to the Merchant. Alice uses non-interactive zero-knowledge arguments to prove:

1. Both representations of p' correspond to the same value and both representations of a correspond to the same value, using the Equivalence of Commitments sub-protocol.
2. $2\sqrt{N} \leq p' < N$, using the Non-equality sub-protocol.
3. $w^{p'} = z (\mathrm{mod} N)$ using the Exponentiation sub-protocol.
4. $h(p') = a$ as a binary string, using the Linear Relations sub-protocol.
5. $a = g_1^{S_A} g_2^{D_A} g_3^{u_1} g_4^{u_2} g_5^{serial} (\mathrm{mod} p)$ using the Exponentiation, Multiplication and Equality sub-protocols. This also proves that a in fact belongs to G_q.
6. $v = D_A + cu_1 (\mathrm{mod} q)$.

Deposit: The merchant sends the transcript of the payment protocol execution to the bank and the bank checks its correctness. The bank checks that *serial* has not been spent before and then credits the merchant's account. If the same payment transcript is deposited twice the bank knows that the merchant tries to deposit the same coin twice. Otherwise if there are two different transcripts for the same money, they both come with the same value *serial* and reveal two different linear equations $r = D_A + cu$ and $r' = D_A + c'u$ in the field Z_q. The bank solves the system of two linear equations to find out D_A which identifies the double spender.

6.2 Security of the off–line payment system

The described payment system can be audited in the same way as the system described in [21], if the factors of the RSA modulus N are unknown (e.g. destroyed during system set up). It further allows to invalidate coins that were withdrawn in a standard (or non-standard) withdrawal session, which is an effective defense for most blackmailing scenarios. Since this discussion is completely analogous to the one in [21] we refer the reader directly to [21] for details and proofs.

Theorem 4. *Under the strong RSA assumption, the DLOG assumption, and under the random oracle assumption, the system is unforgeable and allows to detect double–spenders. Single–spenders have unconditional anonymity. If a user double spends then his identity is revealed, but no knowledge is gained about his secret key S_A. If, in addition, Alice is required to sign each interaction during withdrawal, then no polynomial time bank can falsely accuse her of double spending she has not done. If in addition p_1 and p_2 are not available (e.g., destroyed) after $N = p_1 p_2$ has been generated, then the system is auditable.*

Proof.

Unforgeability: By Theorem 1 we know that if A can prove properties (1-4) for a then a is in the list. Therefore, any spent coin was withdrawn from the bank before.

Anonymity: If a user spends each coin once, then the information that the bank gets to learn includes: T at withdrawal time, $serial, v$ and c at spending time and proofs (arguments) of knowledge. The proofs of knowledge do not reveal any information (in an information theoretical sense).

We next observe that at withdrawal time the bank who sees T has no clue as to the actual representation $T = g_3^{u_1} g_4^{u_2} g_5^{serial}$ the user has for it. At payment time, the user reveals $serial, c$ and $v = D_A + cu_1$. Thus, T that contains u_2 is independent of the information given at payment time.

We are now left to check whether what is sent at payment time reveals any information. Properties (1-5) the user proves are true for any honest transaction. We are left with the value $v = D_A + cu_1$. However, since u_1 is uniform over Z_q, so does v, hence v does not reveal information.

The secret key is protected: The secret key S_A is protected even when a user double spends. To see this notice that the only place a user Alice uses her knowledge of S_A is in the payment protocol where she gives a proof of knowledge of a representation. However, this unconditionally secure proof (argument) of knowledge, provably does not reveal any information about the actual representation Alice knows. All the rest can be simulated with the knowledge of P_A, D_A alone, and the bank can simulate it itself. Thus the bank does not get any information that it could not have obtained from P_A itself.

Double spending: First, because of the unforgeability property, when a user Alice double spends she uses a coin a that has been withdrawn before, let us say w.l.o.g. again by Alice. As all players (including the bank) are polynomial time players they can not find two different representations for any number in G_q (unless with negligible probability) and in particular Alice knows at payment time at most one representation $(a_1, a_2, a_3, a_4, a_5)$ of a with respect to the generators $(g_1, g_2, g_3, g_4, g_5)$. Thus, when Alice double spends a she must use the same representation (or this could be used to extract discrete logs). In particular, she must use *serial* twice, and the bank can identify that these two transactions belong to the same coin.

If Alice convinces the merchant at payment time, then by the soundness property of the proof of knowledge protocol Alice has to reveal (except with negligible probability) the value $a_2 + ca_3$. Now, if Alice double spends, the same coin appears in two payment transcripts with two different linear equations, and Alice must use the same a_2 and a_3 in both cases, because she can not find two different representations for a. Hence she reveals $a_2 = D_A$.

Framing–freeness: If the bank claims Alice double spent a, it has to present the protocol where Alice withdrew a. Therefore, if the bank claims Alice double spent a, then indeed Alice withdrew a and Alice knows a representation $(S_A, a_2, \ldots, a_5)$ of a. As we assume the bank is also polynomial time the bank can not know any other representation for a.

We already proved that S_A is protected even when Alice double spends. That means that the bank gains no information at all about S_A. Thus, if the bank has to answer a random challenge (like the one it gets under the random oracle assumption) then with overwhelming probability the bank can not prove that it knows a representation of a in the basis $(g_1, \ldots, g_k)$. Therefore, a polynomial time bank can not frame Alice for double spending she has not done.

Auditability: We assume that p_1 and p_2 are not known (destroyed after computing $N = p_1 p_2$). An auditor who has access to the public data, i.e. to the accumulated hash value, can easily verify that the bank actions are valid, i.e., given an element a the bank indeed finds a large prime in the set $h^{-1}(a)$, and the list is hashed to the right value, etc.

We note that if, however, p_1 and p_2 are not destroyed, then membership proofs can be given for elements not in the list by those parties knowing the factors, and the system is not auditable.

7 Acknowledgments

The authors are grateful to David Zuckerman for simplifying Algorithm 2.

References

1. N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. *Lecture Notes in Computer Science*, 1233, 1997.
2. M. Bellare and P. Rogaway. Random oracles are practical: A pardigm for designing efficient protocols. In Victoria Ashby, editor, *1st ACM Conference on Computer and Communications Security*, Fairfax, Virginia, November 1993. ACM Press. also appeared as IBM RC 19619 (87000) 6/22/94.
3. J. Benaloh and M. 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.
4. D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In Burt Kaliski, editor, *Advances in Cryptology: CRYPTO '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 425–439. Springer, 1997.
5. S. Brands. An efficient off-line electronic cash system based on the representation problem. In *246*. Centrum voor Wiskunde en Informatica (CWI), ISSN 0169-118X, December 31 1993. AA (Department of Algorithmics and Architecture), CS-R9323, URL=ftp://ftp.cwi.nl/pub/CWIreports/AA/CS-R9323.ps.Z.
6. J. Camenisch and M. Michels. A group signature scheme with improved efficiency. *Lecture Notes in Computer Science*, 1514, 1998.
7. J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. *Lecture Notes in Computer Science*, 1592, 1999.
8. J. L. Carter and M. N. Wegman. Universal classes of hash functions (extended abstract). In *Conference Record of the Ninth Annual ACM Symposium on Theory of Computing*, pages 106–112, Boulder, Colorado, 2–4 May 1977.
9. D. Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology: Proceedings of Crypto 82*, pages 199–203. Plenum Press, New York and London, 1983, 23–25 August 1982.
10. J. D. Cohen and M. J. Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). In *26th Annual Symposium on Foundations of Computer Science*, pages 372–382, Portland, Oregon, 21–23 October 1985. IEEE.
11. 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*. ACM Press, 1999.
12. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew Michael Odlyzko, editor, *Advances in cryptology: CRYPTO '86: proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 181–187, Berlin, 1987. Springer-Verlag.
13. Y. Frankel, P. MacKenzie, and M. Yung. Robust efficient distributed RSA-Key generation. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC-98)*, pages 663–672, New York, May 23–26 1998. ACM Press.

14. Y. Frankel, Y. Tsiounis, and M. Yung. "Indirect discourse proofs": Achieving efficient fair off-line E-cash. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology—ASIACRYPT '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 286–300, Kyongju, Korea, 3–7 November 1996. Springer-Verlag.
15. M. K. Franklin and M. Yung. Secure and efficient off-line digital money (extended abstract). In Svante Carlsson Andrzej Lingas, Rolf G. Karlsson, editor, *Automata, Languages and Programming, 20th International Colloquium*, volume 700 of *Lecture Notes in Computer Science*, pages 265–276, Lund, Sweden, 5–9 July 1993. Springer-Verlag.
16. E. Fujisaki and T. 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.
17. R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. *Lecture Notes in Computer Science*, 1592, 1999.
18. 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.
19. D. Pointcheval and J. Stern. Security proofs for signature schemes. In Ueli Maurer, editor, *Advances in Cryptology—EUROCRYPT 96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 12–16 May 1996.
20. T. Sander. Efficient accumulators without trapdoor. In V. Varadharajan and Y. Mu, editors, *Proceedings of 2nd International Conference on Information and Communication Security (ICICS '99)*, volume 1726 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.
21. T. Sander and A. Ta-Shma. Auditable, anonymous electronic cash. In M.Wiener, editor, *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.
22. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely (extended summary). In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 522–533, Montréal, Québec, Canada, 23–25 May 1994.
23. A. Shamir. On the generation of cryptographically strong pseudo-random sequences. In Shimon Even and Oded Kariv, editors, *Automata, Languages and Programming, 8th Colloquium*, volume 115 of *Lecture Notes in Computer Science*, pages 544–550, Acre (Akko), Israel, 13–17 July 1981. Springer-Verlag.
24. S. von Solms and D. Naccache. On blind signatures and perfect crimes. *Computers and Security*, 11(6):581–583, October 1992.

A Proof of Lemma 1.

Let $H = \{h : \{0,1\}^k \to \{0,1\}^m\}$ be a 2-universal hash family. For $A \subseteq \{0,1\}^k$, $z \in \{0,1\}^m$, we say h is (A,z)-balanced if $0.5 \cdot \frac{|A|}{2^m} \leq |h^{-1}(z) \cap A| \leq 1.5 \cdot \frac{|A|}{2^m}$.

Lemma 2. $\Pr_{h \in H}(h \textit{ is not } (A,z) \textit{ balanced}) \leq O(\frac{2^m}{|A|})$.

Proof. Suppose $A = \{a_1, \ldots, a_l\}$. We pick h uniformly at random from H and let A_i denote the event that $h(a_i) = z$.

Claim. $A_1, \ldots, A_l$ are pairwise independent.

Proof. For every $1 \leq i < j \leq l$,

$$\begin{aligned}\Pr(A_i \cap A_j) &= \Pr_{h \in H}(h(a_i) = z \cap h(a_j) = z) = 2^{-2m} \\ &= \Pr_{h \in H}(h(a_i) = z) \cdot \Pr_{h \in H}(h(a_j) = z) \\ &= \Pr(A_i) \cdot \Pr(A_j).\end{aligned}$$

Where the first equality is by definition, and the second and the third because H is 2-universal. This proves the claim.

Let X_i be one if A_i happens, zero otherwise. Let $X = \Sigma_{i=1}^{l} X_i$, and $\mu = \mathbf{E}(X)$. Then, $X = |h^{-1}(z) \cap A|$ and $\mathbf{E}(X_i) = \Pr_{h \in H}(h(a_i) = z) = 2^{-m}$, so $\mu = l2^{-m}$. By Chebychev:

$$\Pr_{h \in H}(|X - \mu| \leq \frac{\mu}{2}) \leq \frac{4Var(X)}{\mu^2}$$

Now, $X_1, \ldots, X_l$ are pairwise independent, hence $Var(X) = \Sigma Var(X_i) = \Sigma \mathbf{E}(X_i^2) - (\mathbf{E}(X_i))^2 \leq l2^{-m}$. Hence, except for $\frac{4Var(X)}{\mu^2} = O(\frac{2^m}{l})$ fraction of hash functions h, we have:

$$0.5 \cdot \frac{|A|}{2^m} \leq |h^{-1}(z) \cap A| \leq 1.5 \cdot \frac{|A|}{2^m}$$

as required. ■

Let $A \subseteq \{0,1\}^k$ be a subset of $\{0,1\}^k$. For $h \in H$ and $z \in \{0,1\}^m$, we say the pair (h, z) is "bad" for A if $\Pr_{x \in h^{-1}(z)}(x \in A) \leq \frac{\rho(A)}{2}$ where $\rho(A) = \frac{|A|}{2^k}$.

Lemma 3. *For any $A \subseteq \{0,1\}^k$, $z \in \{0,1\}^m$, $\Pr_{h \in H}((h,z)$ is bad for $A) \leq O(\frac{2^m}{|A|})$.*

Proof. Fix $z \in \{0,1\}^k$. When we plug the set $\{0,1\}^k$ into Lemma 2 we see that except for an $O(2^{m-k})$ fraction of the h's,

$$0.5 \cdot 2^{k-m} \leq |h^{-1}(z)| \leq 1.5 \cdot 2^{k-m} \tag{1}$$

Now, let A be an arbitrary subset of $\{0,1\}^k$. By Lemma 2 again, we see that except for an $O(\frac{2^m}{|A|})$ fraction of the h's,

$$0.5 \cdot |A|2^{-m} \leq |h^{-1}(z) \cap A| \leq 1.5 \cdot |A|2^{-m} \tag{2}$$

For any $h \in H$ for which both Equation (1) and Equation (2) hold, we get:

$$\begin{aligned}\Pr_{x \in h^{-1}(z)}(x \in A) = \frac{|h^{-1}(z) \cap A|}{|h^{-1}(z)|} &\geq \frac{0.5 \cdot |A|2^{-m}}{1.5 \cdot 2^{k-m}} \\ &\geq \Omega(\frac{|A|}{2^k}) = \Omega(\rho(A))\end{aligned}$$

and similarly $\Pr_{x \in h^{-1}(z)}(x \in A) \leq O(\rho(A))$. ■

Now, using the union bound we get Lemma 1.