On the Amortized Complexity of Zero-Knowledge Protocols

Ronald Cramer* and Ivan Damgård**

CWI/Leiden University and University of Aarhus

Abstract. We propose a general technique that allows improving the complexity of zero-knowledge protocols for a large class of problems where previously the best known solution was a simple cut-and-choose style protocol, i.e., where the size of a proof for problem instance x and error probability 2^{-n} was $O(|x|n)$ bits. By using our technique to prove n instances simultaneously, we can bring down the proof size per instance to $O(|x|+n)$ bits for the same error probability while using no computational assumptions. Examples where our technique applies include proofs for quadratic residuosity, proofs of subgroup membership and knowledge of discrete logarithms in groups of unknown order, and proofs of plaintext knowledge for various types of homomorphic encryptions schemes. The generality of our method stems from a somewhat surprising application of black-box secret sharing schemes.

1 Introduction

In a zero-knowledge protocol, a prover tries to convince a skeptical verifier that a certain statement is true. Except with a small error probability, the verifier should be convinced if and only the statement is indeed true, but should learn nothing beyond the validity of the assertion. The statement can take the form of claiming that the input string x is in a given language L (interactive proofs) or claiming that the prover knows a "witness" w such that (x, w) is in some given relation R (interactive proofs of knowledge).

Zero-knowledge was introduced in [10], and has been the subject of intense research ever since. Zero-knowlegde protocols are interesting as theoretical objects in their own right, but are also very useful as building blocks in larger protocols.

The efficiency of zero-knowledge proofs have been studied in many works, and in some cases, extremely efficient zero-knowledge proofs have been found. For NP complete problems such as Boolean circuit satisfiability Ishai et al. [12] show protocols where the proof size (communcation complexity) is $O(|x| + poly(k))$ where k is a security parameter and $|x|$ is the size of the input x. In prime order groups, Schnorr's protocol[13] proves knowledge of a discrete logarithm using a (honest verifier) zero-knowledge proof of size $O(|x| + n)$ for an error probability

* Supported by NWO VICI.

** Supported by the Danish Strategic Research Council.

S. Halevi (Ed.): CRYPTO 2009, LNCS 5677, pp. 177–191, 2009.

of 2^{-n}. In a practical application, one must expect to have to communicate x to make the claim in the first place, so this is essentially optimal[1].

However, for several interesting problems, such methods for improved zero-knowledge protocols do not work. This includes, for instance, the very first problem for which a zero-knowledge protocol was suggested, namely quadratic residuosity, where one proves on input x, N that x is a square modulo N (and that the prover knows a square root). The well-known classical protocol for this has error probability $1/2$, and one must repeat it n times for an error probability of 2^{-n} so that the proof will be of size $O(|x|n)$. No more efficient solution with unconditional soundness and zero-knowledge was previously known.

The state of affairs is similar for the discrete log problem in groups of unknown order. Say we are given $g, h \in Z_N^*$ for an RSA modulus N, and the prover claims that h is in the group generated by g, and that he knows the discrete logarithm of h base g. The best solution we know for this has error probability $1/2$, and again we must repeat the entire protocol to reduce the error. Schnorr's protocol cannot be used here since its proof of soundness requires that the group order is known, and finding the order of Z_N^* is equivalent to factoring N. Even if we were happy with only a proof of membership in the group generated by g, the error probability for known solutions would be 1 divided by the smallest prime factor in the total group order, which is $1/2$ for the case of Z_N^*. It should be noted that Fujisaki and Okamoto[8] have shown how to get around these difficulties, but only if we are guaranteed to be working in a subgroup of Z_N^* with only large prime factors in the order, and then only under the strong RSA assumption.

Other examples of a similar nature come from various proposals for homomorphic encryption where one uses subgroups of Z_N^*, often with small prime factors in their order, to make the decryption algorithm efficient[11,6]. In many applications, one needs a zero-knowledge proof that one knows the plaintext for a given ciphertext, and we then have all the same problems as described above.

In this paper, we show a general method that applies to all the problems mentioned above, and allows us to reduce the proof size in the amortized sense: we can give a proof for n instances of the problem simultaneously such that the communication complexity per instance proved is $O(|x| + n)$ for an error probability of 2^{-n}, thus we are as efficient as the best known unconditional protocols for any problem. The technique uses no computational assumptions. In all cases, the computational complexity is also reduced compared to naive repetition of the basic protocol. Here, the most favorable case is discrete log where computation is reduced by a factor n, for quadratic residuosity we gain a factor $\log n$.

We emphasize that for the case of proofs for quadratic residues, what we achieve is different from what can be done using Fiat-Shamir type protocols [9], although they may seem superficially similar to ours: the Fiat-Shamir protocol is also efficient, it takes n quadratic residues as input and has an error probability of 2^{-n}. The difference lies in the type of witness that the prover proves knowledge of. For Fiat-Shamir, the prover only has to know a product of *some*

[1] Although the proof itself could in principle be even smaller if the associated NP-witness is smaller than x.

of the square roots of the input numbers. In fact, the prover could know nothing about the first input number and still survive with probability 1/2. In contrast, our protocol guarantees that the prover knows all the square roots. It should also be noted that the construction of zero-knowledge protocols from multiparty computation in [12] can be adapted to give a protocol for quadratic residuosity with complexity similar to ours. This requires a computational assumption so that, unlike our work, either zero-knowledge or soundness will only be computational. For the other problems we handle in this work, the construction from [12] either leads to larger complexity than ours, or does not not seem to apply at all.

We give an abstract framework characterizing the type of problem that our method applies to, and derive all the examples above as special cases. Basically, we need a function with certain homomorphic properties on some Abelian groups and a ring A that acts on the groups in a sufficiently nice way. The generality of our method comes from a somewhat surprising application of a new result we show for Black-Box Secret-Sharing. This result allows us to use the integers Z as the ring A, and since any Abelian group is a Z-module, we immediately get a general result.

Applications of our result include multiparty computation based on homomorphic encryption, where players would supply inputs by sending them in encrypted form. To make the overall protocol secure, players must prove that they know the inputs they supply, and our method can be used to give such proofs efficiently for a large number of ciphertexts. Note that some computations, such as certain auctions, do in fact require players to submit large amounts of data as input. Another application involves proofs of negative statements such as proving that a number x is not a square modulo N. The classical protocol for this from [10] uses the proof for quadratic residuosity as a subroutine and has complexity $O(|x|n^2)$. Our method reduces this to $O(|x|n)$ without making any computational assumptions. The same idea can used to prove for some homomorphic encryption schemes that a ciphertext contains a non-zero plaintext. Note that these application are for a single instance proof and so are not of an amortized nature.

Finally, we note that our construction generally leads to protocols that are only honest-verifier zero-knowledge(HVZK), as is the case for Schnorr's protocol. But in this paper we are generally happy with this property since first, it is often sufficient when using a protocol as a building block in a larger construction, and second there are several general techniques that can build zero-knowledge protocols from HVZK ones without significant loss of efficiency. While these techniques typically require a complexity assumption, we believe that the technique by Cramer, Damgård and MacKenzie [2] may lead to a solution with no assumptions, this will be the subject of an upcoming paper.

2 The Basic Idea

The first zero-knowledge proof ever presented was the well known protocol to prove quadratic residuosity. We show here a variant related to Goldwasser-Micali

probabilistic cryptosystem, which we will use as a running example in the following. In this cryptosystem, the public key is an RSA modulus N, and we assume for simplicity that it is chosen such that -1 is not a square modulo N. To encrypt a bit w with randomness s, we compute $E_N(w,s) = (-1)^w s^2 \bmod N$. The encryption function is homomorphic, that is, it has two properties: first $E_N(w,s)E_N(w',s') \bmod N = E_N(w \oplus w', ss' \bmod N)$, and moreover, we can multiply a known plaintext b "into a ciphertext", i.e., we have $E_N(w,s)^b = E_N(wb, s^b)$.

Now consider a scenario where the common input to prover P and verifier V is a pair of numbers N and ciphertext x. Now P claims to know a bit w and $s \in Z_N^*$ such that $x = E_N(w,s)$. The protocol goes as follows:

1. P chooses $r \in \{0,1\}, u \in Z_N^*$ at random and sends $a = E_N(r,u)$ to V.
2. V chooses a bit b at random and sends it to P.
3. P sends $z = r \oplus bw, v = us^b \bmod N$ to V, who accepts if and only if $E_n(z,v) = ax^b \bmod N$, and u, v are in Z_N^*.

It is well known that this protocol is perfect zero-knowledge and has error probability 1/2. The reader can easily verify that completeness, soundness and zero-knowledge of the protocol can be based only on the above homomorphic properties of E_N. While error probability of 1/2 is not sufficient in practice, repeating the protocol n times reduces the error probability to 2^{-n}. However, the size of the entire proof will be roughly n times the size of the problem instance.

In this paper we will be concerned with doing it more efficiently if we are to give a proof for n instances of a problem simultaneously. So say we are given a vector $\mathbf{x} = (x_1, ..., x_n)$ of ciphertexts. If we expand the encryption function in a natural way to vectors by applying it to every entry, we can say that the prover's claim now is that he knows vectors $\mathbf{w}, \mathbf{s}$ such that $E_N(\mathbf{w}, \mathbf{s}) = \mathbf{x}$.

Now, the key idea is to consider $\mathbf{w}$, not just as a bit string, but as *an element in the extension field* $GF(2^n)$. Since addition in $GF(2^n)$ is coordinate-wise xor, the (expanded) encryption function is still homomorphic. We have

$$E_N(\mathbf{w}, \mathbf{s})E_N(\mathbf{w}', \mathbf{s}') = E_N(\mathbf{w} + \mathbf{w}', \mathbf{s}\mathbf{s}'),$$

where $\mathbf{w}+\mathbf{w}'$ is addition in $GF(2^n)$ and $\mathbf{s}'\mathbf{s}$ is multiplication in the direct product $(Z_N^*)^n$. We are also able to multiply an element $e \in GF(2^n)$ "into a ciphertext". We can do this by noticing that if we consider $GF(2^n)$ as a vector space over $GF(2)$, multiplication by e is a linear mapping. Taking E to be the matrix of this mapping, multiplying E on an n-bit vector implements multiplication by e. Using this, we can define $\mathbf{x}^e \in (Z_N^*)^n$, where $\mathbf{x} \in (\mathbf{Z}_\mathbf{N}^*)^\mathbf{n}$, namely the i'th entry in $\mathbf{x}^e$ is

$$(\mathbf{x}^e)_i = \prod_{j=1}^{n} x_j^{E(i,j)} \bmod N,$$

where $E(i,j)$ is interpreted as a 0/1 integer. The reader can easily verify that this gives us:

$$E_N(\mathbf{w}, \mathbf{s})^e = E_N(e\mathbf{w}, \mathbf{s}^e).$$

The upshot of this is that since E_N satisfies the same homomorphic properties as before, when seen as a function for encrypting elements in $GF(2^n)$, we can do a proof of knowledge for plaintexts in $GF(2^n)$ by mimicking the protocol above:

1. P chooses $\mathbf{r} \in \{0,1\}^n, \mathbf{u} \in (Z_N^*)^n$ at random and sends $\mathbf{a} = E_N(\mathbf{r}, \mathbf{u})$ to V.
2. V chooses $e \in GF(2^n)$ at random and sends it to P.
3. P sends $\mathbf{z} = \mathbf{r} + e\mathbf{w}, \mathbf{v} = \mathbf{u}\mathbf{w}^e$ to V, who accepts if and only if $E_n(\mathbf{z}, \mathbf{v}) = \mathbf{a}\mathbf{x}^e$, and all entries in $\mathbf{u}, \mathbf{v}$ are in Z_N^*.

Note that V now chooses between 2^n challenges. In fact one can show that if the prover could answer correctly two different challenges e, e', then from the answers we could efficiently compute valid $\mathbf{w}, \mathbf{s}$. The key reason why this is possible is that $e - e'$ is invertible because $GF(2^n)$ is a field (a detailed proof follows as a special case of the general framework we present below).

Hence this protocol has error probability 2^{-n}. Note, however, that we only send a constant number of "compound" ciphertexts to do the protocol. Hence, compared to iterating the basic protocol n times for all n instances which would be the naive solution, we have saved a factor n in the size of the proof.

3 A Framework

In this section we show that the idea we just outlined is not tied to encryption functions over finite fields. All we really need is a function with certain homomorphic properties on Abelian groups, and a ring that acts in "nice" way on the involved groups. To help understand the framework, we use as running example the protocol from the previous section, and show how it is a special case.

3.1 Set-Up and Assumptions

Consider a function $f : R \times S \to X$, where R, S, X are finite Abelian groups. To make the framework fit with the example instantiations to follow, we will write R additively and S, X multiplicatively.

In what follows, we will always assume that we can sample efficiently from all groups that occur, and compute the group operation and inverses efficiently. We also assume that elements can be communicated in some representation such that membership in the relevant group can be checked efficiently. We assume f is "almost" homomorphic, namely it satisfies the following:

$$f(r,s) \cdot f(r',s') = f(r + r, ss'\delta(r,r')) \text{ and } f(0,s)^{-1} = f(0,s^{-1}) \qquad (1)$$

for all $r, r' \in R, s, s' \in S$ and where $\delta(r,r') \in S$ can be efficiently computed from r, r'.

To connect the framework to the previous example, one may think of $R = Z_2, S = X = Z_N^*$ and $f(r,s) = (-1)^r s^2 \bmod N$, where N is such that -1 is a non-square modulo N. Here, of course, we would have $\delta(r,r') = 1$.

We now assume a commutative ring A with 1 such that R is an A- module (this will be the case if $A = Z$, for instance). We assume that A acts on elements

in X, i.e., given $a \in A, x \in X$ one can efficiently compute a new element $x^a \in X$. In the running example, we will set $A = GF(2)$, so an element $a \in A$ is 0 or 1. We then think of these as the *integers* 0 or 1, in which case $x^a \in Z_N^*$ is well defined.

We assume that the action of A on X respects the structure of A, X to some extent, more precisely, we assume for all $x, y \in X, a, b \in A$ that

$$x^a y^a = (xy)^a, \quad x^0 = 1, \quad x^1 = x, \quad 1^a = 1 \tag{2}$$

To complete this picture, we also need an assumption on expressions of form $x^a x^b, (x^a)^b$:

$$x^a x^b = x^{a+b} f(0, \Delta), \quad (x^a)^b = x^{ab} f(0, \Gamma) \tag{3}$$

for all $x \in X, a, b \in A$, and where Δ, Γ can be efficiently computed from x, a, b.

For our running example, (2) is trivially satisfied. For (3), one has to remember that the addition in the exponent is in $GF(2)$ and so is actually an xor. Therefore, the first conditions is satisfied if we set $\Delta = x$ when $a = b = 1$ and $\Delta = 1$ otherwise. The second condition is satisfied by setting $\Gamma = 1$ always.

We also make an assumption on the way $a \in A$ acts on elements in $Im(f) \subset X$:

$$f(r, s)^a = f(a \cdot r, a(s)) \tag{4}$$

for all $a \in A, r \in R, s \in S$. We make no specific assumptions on $a(s) \in S$, other than it can be computed efficiently from a, s. In our example, (4) is satisfied if we just set $a(s) = s^a \bmod N$ where, as above, we think of a as an integer in the natural way.

In the following, we will consider the direct products A^n, R^n, S^n, X^n for a natural number n. Our final assumption is that there exist a special subset $\Omega_n \subset A^n$ and an efficiently computable mapping ω which for every $e \in \Omega_n$ we outputs a matrix $\omega(e)$ with m rows and n columns and entries in A, where m is some function of n and furthermore for every pair $e, e' \in \Omega_n$ where $e \neq e'$, the matrix $\omega(e) - \omega(e')$ is invertible, i.e., there exists an n by m matrix N such that $N(\omega(e) - \omega(e')) = I_n$. Values $e \in \Omega_n$ will be used a challenges in our protocols to follow, and since the error probability will be $1/|\Omega_n|$, we will be looking for constructions that give us a large Ω_n, preferably of size exponentially large in n. In the following, we will usually use E as shorthand for $\omega(e)$.

Definition 1. *If f, A and ω satisfy all of the above conditions, we say that f is* ZK-friendly with respect to *A and ω.*

In our example, we can set Ω_n to be all of $A^n = GF(2)^n$ and $m = n$. Then for $e \in \Omega_n$, we let $E = \omega(e)$ be the matrix that implements multiplication by e in the field $GF(2^n)$, as in the previous section.

3.2 Notation

We will use $\mathbf{r}, \mathbf{s}$ to denote column vectors of elements in R, respectively S, and $f(\mathbf{r}, \mathbf{s})$ to denote the result of applying f to each coordinate.

$$\mathbf{r} = \begin{pmatrix} r_1 \\ r_2 \\ \cdots \\ r_n \end{pmatrix} \quad \mathbf{s} = \begin{pmatrix} s_1 \\ s_2 \\ \cdots \\ s_n \end{pmatrix} \quad f(\mathbf{r}, \mathbf{s}) = \begin{pmatrix} f(r_1, s_1) \\ f(r_2, s_2) \\ \cdots \\ f(r_n, s_n) \end{pmatrix}$$

Let $\mathbf{x}$ be a vector of elements in X, and M is a matrix with entries in A and m rows and n columns. Then we define:

$$\mathbf{x}^M = \begin{pmatrix} \prod_{i=1}^n x_i^{M[1,i]} \\ \prod_{i=1}^n x_i^{M[2,i]} \\ \cdots \\ \prod_{i=1}^n x_i^{M[m,i]} \end{pmatrix} \quad M(\mathbf{s}) = \begin{pmatrix} \prod_{i=1}^n M[1,i](s_i) \\ \prod_{i=1}^n M[2,i](s_i) \\ \cdots \\ \prod_{i=1}^n M[m,i](s_i) \end{pmatrix}$$

It is straightforward to verify that our assumptions on the action of A on X imply that

$$\mathbf{x}^B \mathbf{x}^C = \mathbf{x}^{B+C} f(0^n, \mathbf{\Delta}), \quad (\mathbf{x}^M)^N = \mathbf{x}^{NM} f(0^n, \mathbf{\Gamma}),$$

$$f(\mathbf{r}, \mathbf{s}) f(\mathbf{r}', \mathbf{s}') = f(\mathbf{r} + \mathbf{r}', \mathbf{s}\mathbf{s}' \delta(\mathbf{r}, \mathbf{r}')), \quad f(\mathbf{r}, \mathbf{s})^M = f(M\mathbf{r}, M(\mathbf{r}, \mathbf{s}))$$

for matrices B, C, M, N, where vectors $\mathbf{\Gamma}, \mathbf{\Delta}, M(\mathbf{r}, \mathbf{s})$ can be efficiently computed from the inputs to the operations, and where the function $\delta : R^n \times R^n \to S^n$ is derived from the original δ in the natural way. Note that 0^n denotes the column vector with n zero-entries.

To compute what $M(\mathbf{r}, \mathbf{s})$ should be, one starts from the fact that $(f(\mathbf{r}, \mathbf{s})^M)_j = \prod_{i=1}^n f(\mathbf{r}, \mathbf{s})_i^{M[j,i]}$ and then use (1) and (2). If f is not 1-1, it may be possible to get different values for $M(\mathbf{r}, \mathbf{s})$ depending on the order in which we compute the product. But this is not a problem, in the following we only ned that we can compute *some* element in $M(\mathbf{r}, \mathbf{s}) \in S^n$ that makes $f(\mathbf{r}, \mathbf{s})^M = f(M\mathbf{r}, M(\mathbf{r}, \mathbf{s}))$ be true.

3.3 Some Σ-Protocols

In this section, we assume throughout that we are given a function f that is ZK-friendly w.r.t. some A, ω, and then show that we can build a number of zero-knowledge protocols, more specifically they will be so-called *Σ-protocols*. A *Σ-protocol for a relation* $R = \{(x, w)\}$ is a 3-move protocol for prover P and verifier V. x is the common input and P gets w as private input. Conversations in the protocol have form (a, e, z) where e is a random challenge sent by V. The standard properties of a Σ-protocol is that it is perfectly complete, honest verifier zero-knowledge and sound in the particular sense that from x and conversations $(a, e, z), (a, e', z')$ where $e \neq e'$, one can efficiently compute w such that $(x, w) \in R$. This implies that the protocol is a proof of knowledge for R according to the

standard definition, with knowledge error 1 divided by the number of possible challenges.

The homomorphic property of f described above already implies that there is a Σ-protocol with error probability $1/2$ for the relation $R = \{(x,(w,s))|\ f(w,s) = x\}$. Namely P sends $a = f(r,u)$ for random r, u, and V asks P to send a preimage of either a or xa. This will called *Protocol 0* in the following.

We now give a Σ-protocol for a set of n instances, where the public input is $\mathbf{x} \in X^n$, and the prover demonstrates knowledge of $\mathbf{w}, \mathbf{s}$ such that $f(\mathbf{w},\mathbf{s}) = \mathbf{x}$. In other words, a Σ-protocol for the relation $R_f = \{(\mathbf{x},(\mathbf{w},\mathbf{s}))|\ f(\mathbf{w},\mathbf{s}) = \mathbf{x}\}$ The protocol works as follows:

Protocol 1

1. P chooses vectors $\mathbf{r}, \mathbf{u}$ of length m at random and sends $\mathbf{a} = f(\mathbf{r},\mathbf{u})$ to V.
2. V selects a random element $e \in \Omega_n$ and sends it to P.
3. P sends $\mathbf{z} = E\mathbf{w} + \mathbf{r}$ and $\mathbf{v} = E(\mathbf{w},\mathbf{s}) \cdot \mathbf{u} \cdot \delta(E\mathbf{w},\mathbf{r})$ to V.
4. V accepts if and only if $f(\mathbf{z},\mathbf{v}) = \mathbf{x}^E \cdot \mathbf{a}$.

In this protocol, as well as in all the following, the verifier should also check that every communicated group element is in the group it should be in. For the example from the introduction, this translates to checking that numbers communicated are relatively prime to the modulus N.

Lemma 1. *Protocol 1 is a Σ-protocol for R_f, with error probability $1/|\Omega_n|$. The protocol is also an interactive proof that each entry in $\mathbf{x}$ is in $Im(f)$.*

Proof. Completeness is trivial by the homomorphic property of f. For special soundness, we can assume that we have conversations

$$(\mathbf{a}, e, \mathbf{z}, \mathbf{v}), (\mathbf{a}, e', \mathbf{z}', \mathbf{v}'),\ \text{such that}\ f(\mathbf{z},\mathbf{v}) = \mathbf{x}^E \cdot \mathbf{a},\ \mathbf{f}(\mathbf{z}',\mathbf{v}') = \mathbf{x}^{\mathbf{E}'} \cdot \mathbf{a}$$

and we must compute a valid witness for $\mathbf{x}$. Dividing one equation by the other and using our assumptions a few times, we can conclude that

$$f(\mathbf{z} - \mathbf{z}', \mathbf{v} \cdot \mathbf{v}'^{-1}) = \mathbf{x}^{E-E'} \cdot f(0^n, \mathbf{\Delta})$$

for some $\mathbf{\Delta}$ we can compute efficiently. Setting $A = E - E'$ and moving $f(0^n, \mathbf{\Delta})$ to the other side, we see that we can efficiently compute $\mathbf{c}, \mathbf{d}$ and invertible A such that

$$f(\mathbf{c},\mathbf{d}) = \mathbf{x}^A$$

We then apply the inverse N on both sides, and get

$$f(N \cdot \mathbf{c}, N(\mathbf{c},\mathbf{d})) = (\mathbf{x}^A)^N = \mathbf{x} \cdot f(0^n, \mathbf{\Gamma})$$

for an easily computable vector $\mathbf{\Gamma}$. Moving $f(0^n, \mathbf{\Gamma})$ to the other side, we can easily write $\mathbf{x}$ as $f(\mathbf{r},\mathbf{s})$ for known $\mathbf{r}, \mathbf{s}$, and so we have the required witness. Since we always obtain something in the preimage of x under f, soundness as a proof of membership follows as well.

Finally, we have to provide an honest verifier simulator. For this, we simply choose $e, \mathbf{z}, \mathbf{v}$ uniformly in their respective domains and let $\mathbf{a} = f(\mathbf{x}, \mathbf{v}) \cdot (\mathbf{x}^E)^{-1}$. This clearly simulates the real conversations perfectly, since $\mathbf{z}, \mathbf{v}$ are indeed uniform in real conversations, and $\mathbf{a}$ is fixed when given $\mathbf{z}, \mathbf{v}$.

An straightforward specialization of Protocol 1 can be used to show that $\mathbf{x} = f(0^n, \mathbf{s})$:

Protocol 1.5

1. P chooses vector $\mathbf{u}$ of length m at random and sends $\mathbf{a} = f(0^n, \mathbf{u})$ to V.
2. V selects a random element $e \in \Omega_n$ and sends it to P.
3. P sends $\mathbf{v} = E(0^n, \mathbf{s}) \cdot \mathbf{u} \cdot \delta(0^n, 0^n)$ to V.
4. V accepts if and only if $f(0^n, \mathbf{v}) = \mathbf{x}^E \cdot \mathbf{a}$.

Lemma 2. *Protocol 1.5 is a Σ-protocol for the relation $\{(\mathbf{x}, \mathbf{s}) |\ f(0^n, \mathbf{s}) = \mathbf{x}\}$.*

One immediate generalization of Protocol 1 assumes we have two functions f, g that both satisfy our assumptions for the same A, R, S. We can then build a Σ-protocol for the relation $R_{f,g} = \{(\mathbf{x}, \mathbf{x}', (\mathbf{w}, \mathbf{s}, \mathbf{s}')) |\ f(\mathbf{w}, \mathbf{s}) = \mathbf{x}, g(\mathbf{w}, \mathbf{s}') = \mathbf{x}'\}$, i.e., the demand is that the same $\mathbf{w}$ appears in both preimages. The protocol works as follows:

Protocol 2

1. Start two instances of Protocol 1, using as input $\mathbf{x}$ respectively $\mathbf{x}'$. The prover sends $\mathbf{a}, \mathbf{a}'$, computed using the same value of $\mathbf{r}$ in both instances.
2. The verifier sends one challenge e that the prover uses in both instances to compute the answer.
3. The prover sends $\mathbf{z}, \mathbf{v}, \mathbf{z}', \mathbf{v}'$, and the verifier accepts if and only if $\mathbf{z} = \mathbf{z}'$ and $f(\mathbf{z}, \mathbf{v}) = \mathbf{x}^E \cdot \mathbf{a}$, $g(\mathbf{z}', \mathbf{v}') = {\mathbf{x}'}^E \cdot \mathbf{a}'$.

By following through the proof for Protocol 1, one trivially obtains

Lemma 3. *Protocol 2 is a Σ-protocol for $R_{f,g}$, with error probability $1/|\Omega_n|$. The protocol is also an interactive proof that each entry in $\mathbf{x}$ is in $Im(f)$ and each entry in $\mathbf{x}'$ is in $Im(g)$.*

Protocols assuming R is a ring. We now show that our framework can also be used to show multiplicative relations among preimages under f. To do this, we need to assume that the (additive) group R is actually a ring, and furthermore that we can define an action of R on X and S such that (2), (3), and (4) are satisfied also if we choose $a, b \in R$.

This allows us to define for $x \in Im(f)$ a function

$$f_x(r, s) := x^r \cdot f(0, s)$$

One sees that f_x is almost a homomorphism in the same sense as f, our assumptions immediately imply that we have $f_x(r, s) f_x(r', s') = f_x(r + r', ss' \delta_x(r, r'))$, for some $\delta_x(r, r')$ that is easy to compute from x, r, r'.

Now, suppose we have given $x, y, z \in X$ where a prover knows a, b, c, s_a, s_b, s_c such that $x = f(a, s_a), y = f(b, s_b), z = f(c, s_c)$ and where furthermore $c = ab$. Following several previous works, we can express the relation a bit differently so that it becomes something we can prove using essentially just the protocol we have already.

Notice that if we set $s' = s_c \cdot b(s_a)^{-1} \cdot \delta(ab, 0)^{-1}$, then we have

$$f(c, s_c) = f(ab, s_c) = f_x(b, s')$$

We now consider n instances of such a case, but for a single x and we want a Σ-protocol for the relation R_{mult}, defined as:

$$\{((x, \mathbf{y}, \mathbf{z}), (a, \mathbf{b}, \mathbf{c}, s_a, \mathbf{s_b}, \mathbf{s_c}))|\ x = f(a, s_a), \mathbf{y} = f(\mathbf{b}, \mathbf{s_b}), \mathbf{z} = f(\mathbf{c}, \mathbf{s_c}),\ a \cdot \mathbf{b} = \mathbf{c}\}$$

Then the protocol and lemma below follow immediately:

Protocol 3

1. Run Protocol 0 iterated $\log |\Omega_n|$ times on input x (we can afford to do this on a single input, as it will have the same complexity as the next step).
2. Exploiting the fact that $\mathbf{ab} = \mathbf{c}$, the prover computes $\mathbf{s}'$ as above such that $\mathbf{z} = f_x(\mathbf{b}, \mathbf{s}')$.
3. Do protocol 2 on input $\mathbf{y}, \mathbf{z}$ using f, f_x as the functions f, g.

Lemma 4. *Protocol 3 is a Σ-protocol for R_{mult}.*

As a final example, we show that the framework can be used to show a more negative kind of statement. We need to assume that r is uniquely determined from $f(r, s)$, and second that R is a field. Then we can build an interactive proof system for the language $L = \{x|\ x = f(r, s), r \neq 0\}$.

Protocol 4

1. V chooses n-vectors $\mathbf{r} \in R^n, \mathbf{s} \in S^n$ at random, and computes $\mathbf{g} = f_x(\mathbf{r}, \mathbf{s})$. He sends the $\mathbf{g}$ to P.
2. V uses Protocol 1 to show that he knows $\mathbf{r}, \mathbf{s}$ such that $\mathbf{g} = f_x(\mathbf{r}, \mathbf{s})$.
3. If P accepts the proof in the previous step, he computes $\mathbf{r}$ and sends it to V, who accepts if and only if P sent the correct $\mathbf{r}$

Note that P can do the computation i step 3: since if $x = f(w, s)$ for $w \neq 0$, we have $g_i = x^{r_i} f(0, s_i) = f(wr_i, u_i)$ for some u_i. By assumption wr_i is determined from $f(wr_i, u_i)$ and P can divide out w to get r_i. In general$P$ may need large computing power to find wr_i, but in some cases P can have a trapdoor allowing him do to do it efficiently.

On the other hand if $w = 0$, then $\mathbf{g}$ contains no information on $\mathbf{r}$. Neither does the proof given by V, since it is honest verifier zero-knowledge and hence witness indistinguishable. Therefore, the prover can do no better than a random guess, so the error probability is $|R|^{-n}$. Finally, the protocol is easily seen to be zero-knowledge by a standard argument: the simulator uses rewinding of V to extract $\mathbf{r}$ and can then send exactly what the prover would have sent. If $|R|$ is a small constant such as 2, then Protocol 4 gives a way to improve the complexity over the naive solution where V in step 2 uses Protocol 0 to prove he knows $\mathbf{r}$: we only need to send $O(n)$ group elements, rather than n^2.

3.4 Using Black-Box Secret-Sharing in the Framework

Suppose we are given any function f satisfying (1). Note that if we choose $A = Z$, most of the conditions conditions are automatically satisfied, because any Abelian group is a Z-module. In more concrete terms, it always makes sense to multiply a group element by an integer if the group is written additively (or raise it to an integral power if it is written multiplicatively). In fact, one can easily verify that (2), (3) and (4) are always true if we set $A = Z$, so the only missing condition is the existence of the special subset Ω_n in Z^n and the mapping ω.

A construction of such a set follows from the black-box secret-sharing scheme we introduce in the full version of this paper [1]. The construction itself, however, is easy to understand without this background: recall that Ω_n must be a subset of $A^n = Z^n$. We choose Ω_n to be the set of vectors with entries that are 0 or 1, thus Ω_n has size 2^n. We then need to build, from $e \in \Omega_n$, a matrix $\omega(e)$ with n columns and m rows, where we choose $m = 2n - 1$, and where $e \neq e'$ implies that $\omega(e) - \omega(e')$ is invertible. We do this as follows: thinking of e as a column vector, the j'th column of $\omega(e)$ starts with $j - 1$ zeros, followed by e, followed by $n - j$ zeros.

It is straightforward to show that for any two different e, e', indeed $\omega(e) - \omega(e')$ has an inverse N such that $N(\omega(e) - \omega(e'))$ is the identity matrix. One just observes that the matrix $\omega(e) - \omega(e')$ must always be upper triangular with only 1's or -1's on the diagonal. Therefore we have:

Theorem 1. *Any f satisfying (1) is ZK-friendly with respect to Z and ω constructed as above. In particular, the Σ-protocols 1, 1.5, 2, 3 and 4 will have error probability 2^{-n} and communication complexity linear in n.*

To understand where this construction comes from and why it is connected to secret sharing, it is instructive to have a look at the classical protocol for discrete logarithms, where the prover knows w such that $h = g^w$ in some finite group. The prover sends $a = g^r$, the verifier chooses challenge $e = 0$ or 1, and the prover returns $z = r + ew \bmod t$ where t is the order of g. The verifier checks that $g^z = a \cdot h^e$.

One can interpret this protocol as being based on a very simple 2 out of 2 secret sharing scheme, where the secret is w, r is the randomness used for the sharing, and the shares are r and $r + w$. In this language, the protocol is that the prover commits to the randomness for the secret sharing by sending $a = g^r$, and must then reveal the share of the verifiers choice. The verifier's check ensures that the correct share is indeed revealed. On one hand, since 2 shares are enough to reconstruct, we can extract the secret from any prover who can answer 2 different challenges. On the other hand, since one share reveals no information on the secret, we can simulate the protocol without knowing the secret.

If the group order t is public and is a prime, we can instead use the obvious linear 2 out of t secret sharing scheme where there are t shares and the e'th share is $r + ew \bmod t$. If we again build a protocol by asking the prover to commit to

the randomness by sending g^r and then reveal the share of the verifier's choice, we get exactly Schnorr's protocol. From this point of view, the efficiency of this protocol can be explained from the fact that it is based on a 2 out of t secret sharing scheme for a very large t.

Our protocols from the previous section can be interpreted in a similar way, and we if combine this with the idea of using $A = Z$, we can rephrase our goal as follows: our protocols work with secrets that are vectors of elements in some Abelian group. What we want is to construct a 2 out of T secret sharing scheme (where T can hopefully be chosen very large) which works by acting on the secret vector by integer matrices, and where shares are vectors that are hopefully not much longer than the secret vector. Moreover the scheme should work for any Abelian group. What we are asking for is in fact a novel black-box secret sharing scheme, a concept which is explained in the full version of this paper, where we also develop the secret-sharing scheme that underlies the above theorem.

4 Examples

4.1 Quadratic Residuosity

Let N be a composite number, and let y be a non-square mod N. Then we can set $R = GF(2), S = X = Z_N^*, f(r, s) = y^r s^2 \bmod N, A = GF(2)$.

Now, we can let vectors in $A^n = GF(2)^n$ correspond in the standard way to elements in the extension field $GF(2^n)$. Multiplication by an element $e \in GF(2^n)$ is a linear mapping, so we set $m = n$ and let E be the matrix of this mapping. Finally we can set Ω_n to be all of $GF(2)^n$ since any non-zero element in $GF(2^n)$ is invertible. It is straightforward to check that this satisfies all our assumptions in the framework. Protocol 1 above now becomes a proof that the prover knows how to decrypt n ciphertexts in the Goldwasser-Micali cryptosystem.

The computational cost of the protocols are clearly dominated by the cost of computing the action of E on the vector $\mathbf{x}$. Doing this is equivalent to computing n products of various subsets of n given elements in Z_n^*. Using a straightforward variant of the so called 4 Russians algorithm, this can be done using $O(n^2/\log n)$ multiplications modulo N. We therefore have:

Corollary 1. *Protocol 1 instantiated for the quadratic residuosity case is a proof that the prover knows how to decrypt n ciphertexts in the Goldwasser-Micali cryptosystem. It has communication complexity $2n$ elements in Z_N^* plus $2n$ bits, error probability 2^{-n}, and the computational complexity is $O(n^2/\log n)$ multiplications modulo N.*

Note that if we wanted to obtain the same error probability using simple repetition of the standard cut-and-choose protocol, the cost for all n instances would be $2n^2$ group elements plus $2n$ bits and the computational cost $O(n^2)$ multiplications modulo N. Protocol 1.5 instantiated for this case is easily seen to be a proof that n input numbers are all squares modulo N. It may seem that to use this protocol we need that a non-square y is given, to define the function f, but

this is not the case, since we only need to evaluate f on inputs where the first component is 0, and we always have $f(0, s) = s^2 \bmod N$ no matter which y we would use.

Protocol 4 instantiated for this case is a proof that a given number is a non-square modulo N and this improves the complexity of the classical protocol for this problem from [10] by a factor of n. Again, one can verify that we do not need a non-square y given a priori.

Finally, Protocol 3 in this case becomes a protocol proving that encrypted bit a and encrypted bitstrings $\mathbf{b}, \mathbf{c}$ satisfy $a \wedge \mathbf{b} = \mathbf{c}$, where $a \wedge \mathbf{b}$ is the string obtained by taking the and of a and each bit in $\mathbf{b}$.

4.2 Discrete Log in a Group of Unknown Order

Let N be an arbitrary natural number and $g \in Z_N^*$. Then we will set $R = Z$, S to be the trivial group with one element, and $X = Z_N^*$. We then let $f(r, 1) = g^r \bmod N$. We also set $A = Z$. This does not quite satisfy our framework, since R is not finite, but we will fix this shortly.

The construction behind Theorem 1 implies that we can satisfy the conditions in our framework by construct the set Ω_n as the subset of Z^n consisting of binary strings.

In this case, protocol 1 has to be tweaked slightly: instead of choosing $\mathbf{r}$ uniformly in R^n, which does not make sense when R is infinite, we choose the entries as uniform $\log n + 2k$-bit numbers. This choice ensures both that $f(\mathbf{r})$ will be statistically close to uniform in $Im(f)^m$, and that the entries in $\mathbf{z}$ will be statistically close to uniform $\log n + 2k$-bit numbers. This follows from the fact that the entries in $E \cdot \mathbf{w}$ will be at most $\log n + k$-bit numbers.

The protocol now becomes an interactive proof that the input numbers $x_1, .., x_n$ are all in the group generated by g, and it is a proof that the prover knows the discrete logarithms. The protocol will be honest verifier statistical zero-knowledge.

Corollary 2. *Protocol 1 instantiated for the discrete log in Z_n case is an interactive proof that the input numbers $x_1, .., x_n$ are all in the group generated by g, and it is a proof that the prover knows the discrete logarithms. Let k be the bit length of N. Then the communication complexity is $O(kn)$ bits, the error probability 2^{-n}, and the computational complexity is $O(nk + n^2)$ multiplications modulo N.*

If we wanted to obtain error probability 2^{-n} using simple repetition of the standard cut-and-choose protocol, the cost for n instances would be communication $O(n^2 k)$ bits and also $O(n^2 k)$ multiplications modulo N. So we see that if we choose, e.g., $n = k$, our solution saves a factor k in both the communication and computational complexity.

4.3 Homomorphic Encryption

We already mentioned earlier how our technique can be used for the Goldwasser-Micali probabilistic public-key scheme. This generalizes in a very natural way to

encryptions schemes based on higher degree residuosity, say degree q for q a prime larger than 2, provided q divides $\phi(N)$. The plaintext-space for the encryption would be $R = Z_q$ and one would then define the encryption of plaintext r as $f(r, s) = y^r s^q \bmod N$ where y is not a q-power modulo N. The basic Protocol 0 with $A = Z_q$ and $\Omega_n = Z_q^n$ gives a proof of knowlegde of the plaintext for a given ciphertext with error probability $1/q$. Using Protocol 1, this can be amplified to a proof for n plaintexts with error probability q^{-n}, at cost n times the cost of Protocol 0.

In [11], a different type of encryption function is proposed, also based on a composite modulus N and two elements $g, h \in Z_N^*$. The encryption function is $f(m, s) = g^m h^s \bmod N$. Here m is the message chosen in Z_M for a public M and s is chosen at random in some interval $[0..T]$. We do not need to go into the details of the scheme and its security here, it is enough to say that the order of h has to be secret and one needs to assume for security that a random element in the group generated by h cannot be efficiently distinguished from a random element in Z_N^*.

Standard methods for proving in zero-knowledge that you know m, s for a given ciphertext have error probability $1/2$, namely one does the obvious Σ-protocol with a binary challenge. One cannot do better using Schnorr-like techniques because one would need to know the order of h to do the knowledge extraction required for soundness. However, the scheme fits in our framework, by setting $R = Z_M$, $S = Z$, $X = Z_N^*$ and $A = Z$. Now, using Theorem 1, Protocol 1 shows that we can prove knowledge of n plaintexts with error probability 2^{-n} at cost about $2n$ times the standard protocol for a single instance.

Finally, we note that if g has order M, R can act on S and X as required for Protocols 3 and 4. Protocol 3 can be used to show multiplicative relations among plaintexts, and in case the plaintext space is a field (i.e., if M is a prime). Protocol 4 can be used to show that a ciphertext contains a non-zero plaintext.

References

1. Cramer, R., Damgård, I.B.: On the Amortized Complexity of Zero-Knowledge Protocols. Full version of this paper (in preparation)
2. Cramer, R., Damgård, I.B., MacKenzie, P.D.: Efficient Zero-Knowledge Proofs of Knowledge Without Intractability Assumptions. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 354–373. Springer, Heidelberg (2000)
3. Cramer, R., Fehr, S.: Optimal Black-box Secret Sharing over Arbitrary Abelian Groups. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, p. 272. Springer, Heidelberg (2002)
4. Cramer, R., Fehr, S., Stam, M.: Primitive Sets over Number Fields and Black-Box Secret Sharing. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 344–360. Springer, Heidelberg (2005)
5. Damgård, I.B., Ishai, Y.: Scalable Secure Multiparty Computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 501–520. Springer, Heidelberg (2006)
6. Damgård, I.B., Geisler, M., Krøigaard, M.: Efficient and Secure Comparison for On-Line Auctions. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 416–430. Springer, Heidelberg (2007)

7. Desmedt, Y., Frankel, Y.: Homomorphic Zero-Knowledge Threshold Schemes over Any Finite Abelian Group. SIAM J. on Discrete Mathematics 7(4), 667–679 (1994)
8. Fujisaki, E., Okamoto, T.: Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
9. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1985)
10. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof-Systems. In: Proceedings of STOC 1985, pp. 291–304 (1985)
11. Groth, J.: Cryptography in Subgroups of Zn. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 50–65. Springer, Heidelberg (2005)
12. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of STOC 2007, pp. 21–30 (2007)
13. Schnorr, C.-P.: Efficient Signature Generation by Smart Cards. J. Cryptology 4(3), 161–174 (1991)