J. Cryptol. (2014) 27: 284–316
DOI: 10.1007/s00145-013-9145-x

Journal of
CRYPTOLOGY

On the Amortized Complexity of Zero-Knowledge Protocols

Ronald Cramer[*]
CWI, Amsterdam, The Netherlands
cramer@cwi.nl
and
Leiden University, Leiden, The Netherlands
cramer@math.leidenuniv.nl

Ivan Damgård[†]
Aarhus University, Aarhus, Denmark
ivan@cs.au.dk

Marcel Keller[‡]
University of Bristol, Bristol, UK
m.keller@bristol.ac.uk

Communicated by Oded Goldreich

Received 31 October 2011
Online publication 31 January 2013

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, interval proofs of the latter, and proofs of plaintext knowledge for various types of homomorphic encryption schemes. We first propose our protocols as Σ-protocols and extend them later to zero-knowledge proofs of knowledge.

Key words. Σ-protocols, Zero-knowledge, Proof of knowledge, Homomorphic encryption, Random self-reducible problems.

[*] R. Cramer was supported by NWO VICI.

[†] I. Damgård was supported by the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino–Danish Center for the Theory of Interactive Computation, within which part of this work was performed; and also by the CFEM research center (supported by the Danish Strategic Research Council) within which part of this work was performed.

[‡] M. Keller was supported in part by EPSRC via grant EP/I03126X.

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 $\mathcal{R}$ (interactive proofs of knowledge).

Zero-knowledge was introduced in [8] and has been the subject of intense research ever since. Zero-knowledge 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 has 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. [10] show protocols where the proof size (communication complexity) is $O(|x| + \text{poly}(k))$, where k is a security parameter, and $|x|$ is the size of the input x. In prime-order groups, Schnorr's protocol [11] proves knowledge of a discrete logarithm using a (honest verifier) zero-knowledge proof of size $O(|x| + n)$ for an error probability of 2^{-n}. In a practical application, one must expect to have to communicate x to make the claim in the first place as well as an n-bit challenge, 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 [4] 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 $g, h \in \mathbb{Z}_N^*$ 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 $\mathbb{Z}_N^*$ 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 $\mathbb{Z}_N^*$. It should be noted that Fujisaki and Okamoto [7] have shown how to get around these difficulties, but only if we are guaranteed to be working in a subgroup of $\mathbb{Z}_N^*$ 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 $\mathbb{Z}_N^*$, often with small prime factors in their order, to make the decryption algorithm efficient [5,9]. Many applications require a zero-knowledge proof that one knows the plaintext for a given ciphertext, and we then have all the same problems as described above.

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

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 [6], 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 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 also makes sense to compare our work to the "MPC-in-the-head" technique from [10], where the idea is to construct a zero-knowledge protocol from a multiparty computation protocol. The technique requires a commitment scheme, and it is well known that such schemes cannot offer unconditional security for both parties. This means that for any protocol constructed using MPC-in-the-head, either soundness or zero-knowledge will be computational, unlike our results that require no computational assumptions. Nevertheless, it should be noted that one can adapt the construction from [10] to give a protocol for quadratic residuosity modulo some number N with complexity similar to ours. This is because one can start the construction of [10] from a multiparty protocol that is particularly efficient because it is based on a special-purpose secret sharing scheme that works modulo N. For other cases, such as discrete log in a group of unknown order, no such secret-sharing scheme is available, one has to resort to generic MPC protocols, and the complexity then becomes much larger than ours.

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 latter holds if all groups are A-modules (although this is not a necessary condition). It follows that $\mathbb{Z}$ can be always used as A since all Abelian groups are $\mathbb{Z}$-modules.

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, see [3] for a concrete example of such an application. 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 [8] 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 nonzero plaintext. Note that these applications are for a single instance proof and so are not of an amortized nature.

The Σ-protocols presented in Sect. 3.4 are only honest-verifier zero-knowledge (HVZK), as it is the case for Schnorr's protocol. In Sect. 5, we present an extended version of our basic protocol (called Protocol 1 in the following), which is a zero-knowledge proof of knowledge. We use a technique by Bellare et al. [2] to construct it but extend their result by showing soundness as a proof of knowledge, which was not done in [2]. As common for zero-knowledge proofs, the verifier has to commit to his challenge at the beginning of the protocol. Here, the commitment scheme is constructed from basic Σ-protocol for a new instance generated by the prover. The extended protocol has only constant-factor overhead of the basic one.

1.1. *An Intuitive Explanation of Our Technique*

Our basic idea is closely connected to secret sharing (a concept that we assume in the reader is familiar with). To understand this, it is instructive to have a look at the well-known zero-knowledge protocol for discrete logarithms, where the prover claims to know w such that $h = g^w$ in some finite group. The prover sends $a = g^r$, the verifier chooses a 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 two shares are enough to reconstruct, we can extract the secret from any prover who can answer two 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 eth 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.

The protocols from this paper be understood in a similar way: the prover's secret is a vector of witnesses $w_1, \dots, w_n$, and, loosely speaking, the idea is to construct a suitable 2-out-of-2^n secret sharing scheme and otherwise do "the same" as in Schnorr's protocol. We show a concrete example of this in the following section.

2. An Example of Our 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 $s \in \mathbb{Z}_N^*$ such that $x = E_N(w,s)$. The protocol goes as follows:

1. P chooses $r \in \{0,1\}$ and $u \in \mathbb{Z}_N^*$ 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 $z = r \oplus bw$ and $v = us^b \bmod N$ to V, who accepts if and only if $E_n(z,v) = ax^b \bmod N$ and u, v are in $\mathbb{Z}_N^*$.

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 the error probability $1/2$ 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 $\mathbf{x} = (x_1, \dots, x_n)$ 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 $(\mathbb{Z}_N^*)^n$. 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 $\mathbf{x}^e \in (\mathbb{Z}_N^*)^n$, where $\mathbf{x} \in (\mathbb{Z}_N^*)^n$, namely the ith 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 $\mathbf{r} \in \{0, 1\}^n$ and $\mathbf{u} \in (\mathbb{Z}_N^*)^n$ 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 $\mathbf{z} = \mathbf{r} + e\mathbf{w}$ and $\mathbf{v} = \mathbf{u}\mathbf{s}^e$ 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 $\mathbb{Z}_N^*$.

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.

This construction can also be described in the "secret-sharing language" from the introduction: the prover's secret is a pair of vectors $\mathbf{w}$, $\mathbf{s}$, there is a share for each element $e \in GF(2^n)$, and with randomness $\mathbf{r}, \mathbf{u}$ the eth share is computed as $\mathbf{r} + e\mathbf{w}, \mathbf{u}\mathbf{s}^e$.

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 (the encryption function in our example), and a ring that acts in a "nice" way on the involved groups ($GF(2)$ in our example). To help understand the framework, we use the protocol from the previous section as a running example.

3.1. *Setup 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. We require that f is "almost" homomorphic and that X is "almost" a A-module with A being a commutative ring with 1. The following definition explains what we mean by this algebraically.

Definition 1. Let R, S, X be Abelian groups written as above, A a commutative ring with 1, and $f : R \times S \to X$ a function. We say that f is *ZK-ready with respect to A* if all of the following holds.

There exist $g : R \to X$ and a group homomorphism $h : S \to X$ such that

$$\begin{aligned} f(r, s) &= g(r) \cdot h(s) \\ (\pi \circ g)(r + r') &= (\pi \circ g)(r) \cdot (\pi \circ g)(r') \end{aligned} \tag{1}$$

$$g(0) = 1 \tag{2}$$

for all $r, r' \in R$, $s \in S$, where $\pi : X \to X/\operatorname{Im}(h)$ denotes the canonical projection. In other words, $\pi \circ g$ is a group homomorphism.

Every $a \in A$ acts as an endomorphism of X, i.e., given $x \in X$, there exists $x^a \in X$ such that

$$x^a y^a = (xy)^a \tag{3}$$

for all $x, y \in X$. In particular, $0 \in A$ acts as the trivial endomorphism, and $1 \in A$ acts as the identity, i.e.,

$$x^0 = 1, \quad x^1 = x \tag{4}$$

for every $x \in X$.

R and $\operatorname{Im}(\pi)$ are A-modules, $\operatorname{Im}(\pi \circ g)$ is an A-submodule of the latter, $\pi \circ g$ is an A-module homomorphism, and

$$\pi\bigl(x^a\bigr) = \pi(x)^a \tag{5}$$

for all $a \in A$, $x \in X$.

To connect the framework to the example in the previous section, one may think of $R = \mathbb{Z}_2$, $S = X = \mathbb{Z}_N^*$, and $f(r,s) = (-1)^r s^2 \bmod N$, where N is such that -1 is a nonsquare modulo N. Here, of course, we have that $g(r) := (-1)^r$ is a group homomorphism because $(-1)^2 = 1$. Therefore, $\pi \circ g$ is a group homomorphism as well. Furthermore, we set $A = GF(2)$, so an element $a \in A$ is 0 or 1. $x^a \in \mathbb{Z}_N^*$ is then defined by (4), $\operatorname{Im}(\pi) \cong \operatorname{Im}(\pi \circ g) \cong \mathbb{Z}_2$ clearly is a $GF(2)$-module, and (5) is trivially achieved.

Jumping ahead, we mention that if we set $A = \mathbb{Z}$, the above conditions on A are always satisfied, and in fact we show in Theorem 1 below that all conditions in our framework can be satisfied for $A = \mathbb{Z}$ and any f satisfying (1) and (2).

For a practical usage of our framework, we need the following two lemmas. They establish several functions that quantify the deviation of f and X from being a homomorphic function and an A-module, respectively. We will use the equalities stated in the lemmas to prove the properties of our proposed Σ-protocols as one would use the homomorphic property of discrete exponentation and the fact that any cyclic group is a $\mathbb{Z}$-module to prove that Schnorr's protocol is a Σ-protocol.

Lemma 1. *Let f be ZK-ready with respect to A, and let all symbols be as above. Then, there exist $\delta : R \times R \to S$, $\Delta : X \times A \times A \to S$, $\Gamma : X \times A \times A \to S$, $\gamma : A \times R \times S \to S$ such that*

$$f(r,s) \cdot f\bigl(r', s'\bigr) = f\bigl(r + r', ss'\delta\bigl(r, r'\bigr)\bigr) \tag{6}$$

$$\begin{aligned} x^a x^b &= x^{a+b} \cdot f\bigl(0, \Delta(x,a,b)\bigr) \\ \bigl(x^a\bigr)^b &= x^{ab} \cdot f\bigl(0, \Gamma(x,a,b)\bigr) \end{aligned} \tag{7}$$

$$f(r,s)^a = f\bigl(a \cdot r, \gamma(a,r,s)\bigr) \tag{8}$$

for all $r, r' \in R$, $s, s' \in S$, $x \in X$, $a, b \in A$.

In our example, $\delta(r, r') = 1$ for all $r \in R$ because f is homomorphic. For (7), one has to remember that the addition in the exponent is in $GF(2)$ and so is actually an xor. Therefore, $\varDelta(x, a, b) = x$ when $a = b = 1$ and $\varDelta(x, a, b) = 1$ otherwise. On the other hand, $\varGamma(x, a, b) = 1$ always, and $\gamma(a, r, s) = s^a$.

It can be shown that (1) is in fact equivalent to (6). Given (5), one can also show that $\mathrm{Im}(\pi)$ being an A-module is equivalent to (7) and $\mathrm{Im}(\pi \circ g)$ being an A-module is equivalent to (8).

Proof. (1) implies that

$$\begin{aligned}&\pi\big(g(r+r')\cdot g(r)^{-1}\cdot g(r')^{-1}\big) = (\pi\circ g)(r+r'-r-r') = 1\\ \Rightarrow\quad &\big(g(r+r')\cdot g(r)^{-1}\cdot g(r')^{-1}\big) \in \mathrm{Im}\,h,\end{aligned}$$

and thus, the following is well defined for all $r, r' \in R$:

$$\delta(r, r') := h^{-1}\big(g(r+r')^{-1}\cdot g(r)\cdot g(r')\big), \tag{9}$$

where $h^{-1} : \mathrm{Im}(h) \to S$ can be any inverse of h. It directly follows that f satisfies the following:

$$\begin{aligned}f(r,s)\cdot f(r',s') &= g(r)\cdot h(s)\cdot g(r')\cdot h(s')\\ &= g(r+r')\cdot h(ss')\cdot g(r+r')^{-1}\cdot g(r)\cdot g(r')\\ &= g(r+r')\cdot h\big(ss'\delta(r,r')\big)\\ &= f\big(r+r', ss'\delta(r,r')\big)\end{aligned}$$

for all $r, r' \in R,\ s, s' \in S$.

From (5) and the fact that $\mathrm{Im}(\pi)$ is an A-module, it follows that

$$\begin{aligned}\pi\big(x^a x^b\big) = \pi(x)^a\pi(x)^b = \pi(x)^{a+b} = \pi\big(x^{a+b}\big)\\ \pi\big((x^a)^b\big) = \big(\pi(x)^a\big)^b = \pi(x)^{ab} = \pi\big(x^{ab}\big)\end{aligned}$$

for all $x \in X,\ a, b \in A$. Similarly to above, one can therefore define

$$\begin{aligned}\varDelta(x,a,b) &:= h^{-1}\big(x^a x^b \big(x^{a+b}\big)^{-1}\big)\\ \varGamma(x,a,b) &:= h^{-1}\big((x^a)^b\big(x^{ab}\big)^{-1}\big),\end{aligned} \tag{10}$$

and (7) follows.

Finally,

$$\pi\big(f(r,s)^a\big) = \pi\big(g(r)\big)^a\pi\big(h(s)\big)^a = \pi\big(g(a\cdot r)\big)$$

for all $r \in R,\ s \in S$. Therefore, one can define

$$\gamma(a, r, s) := h^{-1}\big(f(r,s)^a\cdot g(a\cdot r)^{-1}\big), \tag{11}$$

which achieves (8). □

Lemma 2. *Let f be ZK-ready with respect to A, and δ as in the previous lemma. Then,*

$$f\big(r, ss'\big) = f(r, s) \cdot f\big(0, s'\big) \tag{12}$$

$$f(r, s)^{-1} = f\big(-r, s^{-1} \cdot \delta(r, -r)^{-1}\big) \tag{13}$$

$$f(0, s)^{-1} = f\big(0, s^{-1}\big) \tag{14}$$

for all $r \in R,\ s \in S$.

This lemma also constitutes the raising d'être of (2), namely to simplify the computations. The framework would also work without it; nevertheless, all known examples achieve the condition.

Proof. The following follows from (1), (2), and (6):

$$f(0, 1) = g(0) \cdot h(1) = 1$$

$$f\big(r, ss'\big) = g(r) \cdot h\big(ss'\big) = g(r) \cdot h(s) \cdot g(0) \cdot h\big(s'\big) = f(r, s) \cdot f\big(0, s'\big)$$

$$f(0, s)^{-1} = g(0)^{-1} h(s)^{-1} = g(0) \cdot h\big(s^{-1}\big) = f\big(0, s^{-1}\big)$$

$$f(r, s) \cdot f\big(-r, s^{-1}\delta(r, -r)^{-1}\big) = f\big(r - r, ss^{-1}\delta(r, -r)\delta(r, -r)^{-1}\big) = f(0, 1) = 1$$

for all $r \in R, s, s' \in S$, and thus,

$$f(r, s)^{-1} = f\big(-r, s^{-1} \cdot \delta(r, -r)^{-1}\big). \qquad \square$$

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 that for every $e \in \Omega_n$ outputs a matrix $\omega(e)$ with m rows and n columns and entries in A, where m is some polynomial 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

$$N\big(\omega(e) - \omega\big(e'\big)\big) = I_n.$$

Values $e \in \Omega_n$ will be used as 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 use E as shorthand for $\omega(e)$.

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.

The following definition combines all requirements for our framework, algebraical and computational.

Definition 2. Let f be ZK-ready with respect to A. We say that f is *ZK-friendly with respect to A and ω* if there exists ω as described above for every natural number n and there is a representation of the groups R, S, X and the ring A such that all of the following is efficiently computable:

- uniformly random elements of all groups (sampling)
- membership check for all groups
- group operations and inversions in all groups
- addition, additive inverse, and multiplication in A
- the action of A on X
- δ, Δ, Γ, and a' as defined in Lemma 1
- ω and N inverting $(\omega(e) - \omega(e'))$ for every distinct pair $e, e' \in \Omega_n$

3.2. *A Generic Challenge Space*

Suppose we are given any function f satisfying (1) and (2). Note that if we choose $A = \mathbb{Z}$, most of the other conditions of Definition 1 are automatically satisfied, because any Abelian group is a $\mathbb{Z}$-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, (3), (4), and (5) follow trivially if we set $A = \mathbb{Z}$, so the only missing condition is the existence of the special subset Ω_n in $\mathbb{Z}^n$ and the mapping ω.

Recall that Ω_n must be a subset of $A^n = \mathbb{Z}^n$. We choose Ω_n to be the set of vectors with entries that are 0 or 1, and 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 jth column of $\omega(e)$ starts with $j - 1$ zeros, followed by e, followed by $n - j$ zeros:

$$\begin{pmatrix} e_1 & & 0 \\ \vdots & \ddots & \\ e_n & & e_1 \\ & \ddots & \vdots \\ 0 & & e_n \end{pmatrix}$$

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 1s or -1s on the diagonal. Therefore we have the following:

Theorem 1. *Any $f : R \times S \to X$ satisfying (1) and (2) is ZK-friendly with respect to $\mathbb{Z}$ and ω constructed as above if R, S, and X fulfill the first three conditions of Definition 2 and if δ as defined in Lemma 1 is efficiently computable.*

Proof. By (10), Δ and Γ are constantly 1, and

$$\begin{aligned}\gamma(a,r,s) &= h^{-1}\bigl(f(r,s)^a \cdot g(a\cdot r)^{-1}\bigr)\\ &= s^a \cdot h^{-1}\bigl(g(r)^a \cdot g(a\cdot r)^{-1}\bigr)\\ &= s^a \cdot h^{-1}\Biggl(\prod_{i=2}^{a}\bigl(g(i\cdot r)^{-1}\cdot g\bigl((i-1)\cdot r\bigr)\cdot g(r)\bigr)\Biggr)\\ &= s^a \cdot \prod_{i=2}^{a}\delta(i\cdot r, r)\end{aligned}$$

by (11) and (9). Furthermore, arithmetic in $\mathbb{Z}$ is efficient, and the action of $\mathbb{Z}$ on X is efficiently computable if group operations in X are. It follows that all requirements in Definition 2 are achieved. □

It is worth noting that this construction of ω is closely connected to secret sharing, in exactly the way we explained in the introduction: if we consider our protocols from the previous section and combine with the idea of using $A=\mathbb{Z}$, we can rephrase the construction as follows: our protocols work with secrets that are vectors of elements in some Abelian group. What we construct is a 2-out-of-2^n secret sharing scheme where the eth share is computed by acting on the secret vector by the matrix $\omega(e)$.

3.3. *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:

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

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

$$\mathbf{x}^M = \begin{pmatrix} \prod_{i=1}^n x_i^{M[1,i]}\\ \prod_{i=1}^n x_i^{M[2,i]}\\ \vdots\\ \prod_{i=1}^n x_i^{M[m,i]}\end{pmatrix}$$

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

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

$$f(\mathbf{r},\mathbf{s}) f\bigl(\mathbf{r}',\mathbf{s}'\bigr) = f\bigl(\mathbf{r}+\mathbf{r}', \mathbf{s}\mathbf{s}'\delta\bigl(\mathbf{r},\mathbf{r}'\bigr)\bigr), \qquad f(\mathbf{r},\mathbf{s})^M = f\bigl(M\mathbf{r}, \gamma(M,\mathbf{r},\mathbf{s})\bigr)$$

for matrices B, C, M, N, where the vectors $\boldsymbol{\Delta}(\mathbf{x}, B, C)$, $\boldsymbol{\Gamma}(\mathbf{x}, M, N)$, $\delta(\mathbf{r}, \mathbf{r}')$, and $\gamma(M, \mathbf{r}, \mathbf{s})$ can be efficiently computed as follows:

$$\begin{aligned}
\big(\boldsymbol{\Delta}(\mathbf{x}, B, C)\big)_j &:= \prod_{i=1}^{n} \Delta\big(x_i, B[j,i], C[j,i]\big)\\
\big(\boldsymbol{\Gamma}(\mathbf{x}, M, N)\big)_j &:= \prod_{i=1}^{n} \prod_{k=1}^{n} \Gamma\big(x_i, M[k,i], N[j,k]\big)\\
\big(\delta\big(\mathbf{r}, \mathbf{r}'\big)\big)_j &:= \delta\big(r_j, r'_j\big)\\
\big(\gamma(M, \mathbf{r}, \mathbf{s})\big)_j &:= \prod_{i=1}^{n} \gamma\big(M[j,i], r_i, s_i\big) \cdot \prod_{i=2}^{n} \delta\left(M[j,i]r_i, \sum_{k=1}^{i-1} M[j,k]r_k\right)
\end{aligned}$$

for all j. Note that 0^n denotes the column vector with n zero entries.

To compute what $\gamma(M, \mathbf{r}, \mathbf{s})$ 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 (3), (6), and (7). If f is not 1–1, it may be possible to get different values for $\gamma(M, \mathbf{r}, \mathbf{s})$ depending on the order in which we compute the product. But this is not a problem since, in the following, we only need that we can compute *some* element in $\gamma(M, \mathbf{r}, \mathbf{s}) \in S^n$ that makes $f(\mathbf{r}, \mathbf{s})^M = f(M\mathbf{r}, \gamma(M, \mathbf{r}, \mathbf{s}))$ be true.

3.4. *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* $\mathcal{R} = \{(x, w)\}$ 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 are 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 $(x, w) \in \mathcal{R}$. This implies that the protocol is a proof of knowledge for $\mathcal{R}$ 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 $\mathcal{R} = \{(x, (w, s)) \mid f(w, s) = x\}$. 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 be 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 $\mathcal{R}_f = \{(\mathbf{x}, (\mathbf{w}, \mathbf{s})) \mid f(\mathbf{w}, \mathbf{s}) = \mathbf{x}\}$. 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 $\mathbf{v} = \gamma(E, \mathbf{w}, \mathbf{s}) \cdot \mathbf{u} \cdot \delta(E\mathbf{w}, \mathbf{r})$ 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 3. *Protocol* 1 *is a $\varSigma$-protocol for $\mathcal{R}_f$, with error probability* $1/|\varOmega_n|$. *The protocol is also an interactive proof that each entry in* $\mathbf{x}$ *is in* $\operatorname{Im}(f)$.

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

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

and we must compute a valid witness for $\mathbf{x}$. Dividing one equation by the other and applying (13) as well as the definition of $\boldsymbol{\varDelta}$ gives

$$f(\mathbf{z}, \mathbf{v}) \cdot f\bigl(-\mathbf{z}', \mathbf{v}'^{-1} \cdot \delta\bigl(\mathbf{z}', -\mathbf{z}'\bigr)^{-1}\bigr) = \mathbf{x}^{E-E'} \cdot f\bigl(0^n, \boldsymbol{\varDelta}\bigl(\mathbf{x}, E, -E'\bigr)\bigr),$$

which is by (6), (12), and (14) equivalent to

$$f\bigl(\mathbf{z} - \mathbf{z}', \mathbf{v} \cdot \mathbf{v}'^{-1} \cdot \delta\bigl(\mathbf{z}', -\mathbf{z}'\bigr)^{-1} \cdot \delta\bigl(\mathbf{z}, \mathbf{z}'\bigr) \cdot \boldsymbol{\varDelta}\bigl(\mathbf{x}, E, -E'\bigr)^{-1}\bigr) = \mathbf{x}^{E-E'}.$$

Defining $\mathbf{c}, \mathbf{d}$ as the arguments of f on the right-hand side and $M := E - E'$ gives

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

We then apply the inverse N of M, which exists by Definition 2, on both sides and get

$$f\bigl(N \cdot \mathbf{c}, \gamma(N, \mathbf{c}, \mathbf{d})\bigr) = \bigl(\mathbf{x}^M\bigr)^N = \mathbf{x} \cdot f\bigl(0^n, \boldsymbol{\varGamma}(\mathbf{x}, M, N)\bigr).$$

Applying (12) and (14) once more, we conclude

$$f\bigl(N\mathbf{c}, \gamma(N, \mathbf{c}, \mathbf{d}) \cdot \boldsymbol{\varGamma}(\mathbf{x}, M, N)^{-1}\bigr) = \mathbf{x},$$

and so we have the required witness. Since we always obtain something in the preimage of $\mathbf{x}$ 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 $\mathbf{a} = f(\mathbf{z}, \mathbf{v}) \cdot (\mathbf{x}^E)^{-1}$. 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 $\mathbf{z}, \mathbf{v}$. □

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

Protocol 1.5.

1. P chooses a 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 $e \in \varOmega_n$ and sends it to P.

3. P sends $\mathbf{v} = \gamma(E, 0^n, \mathbf{s}) \cdot \mathbf{u}$ to V.
4. V accepts if and only if $f(0^n, \mathbf{v}) = \mathbf{x}^E \cdot \mathbf{a}$.

Lemma 4. *Protocol 1.5 is a Σ-protocol for the relation* $\{(\mathbf{x}, \mathbf{s}) \mid f(0^n, \mathbf{s}) = \mathbf{x}\}$.

One immediate generalization of Protocol 1 assumes that 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 $\mathcal{R}_{f,g} = \{(\mathbf{x}, \mathbf{x}', (\mathbf{w}, \mathbf{s}, \mathbf{s}')) \mid f(\mathbf{w}, \mathbf{s}) = \mathbf{x}, g(\mathbf{w}, \mathbf{s}') = \mathbf{x}'\}$, 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 the following lemma.

Lemma 5. *Protocol 2 is a Σ-protocol for $\mathcal{R}_{f,g}$, with error probability $1/|\Omega_n|$. The protocol is also an interactive proof that each entry in $\mathbf{x}$ is in $\mathrm{Im}(f)$ and each entry in $\mathbf{x}'$ is in $\mathrm{Im}(g)$.*

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 such that (3), (4), and (5) are satisfied for $A = R$. We also need that the multiplication operation of R agrees with operation of R on $\mathrm{Im}(\pi \circ g)$, i.e.,

$$\pi\big(g(r')\big)^r = \pi\big(g(rr')\big)$$

for all $r, r' \in R$, where π and g are induced by f as in Definition 1.

Then, we define for $x \in \mathrm{Im}(f)$ a function

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

for h induced by f as in Definition 1. If $\pi(x^r) = \pi(g(r))$ for all $r \in R$, $\pi \circ g_x = \pi \circ g$ clearly achieves (1).

Now, suppose we are 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 $s' = s_c \cdot \gamma(b, a, s_a)^{-1} \cdot \delta(ab, 0)^{-1}$, then we have

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

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

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

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 6. *Protocol* 3 *is a Σ-protocol for $\mathcal{R}_{mult}$ with error probability* $1/|\Omega_n|$.

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 $L=\{x\mid x=f(r,s),r\neq 0\}$.

Protocol 4.

1. V chooses n-vectors $\mathbf{r}\in R^n$ and $\mathbf{s}\in S^n$ 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 in step 3: since $x=f(w,s)$ 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 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.

The following corollary applies Theorem 1 to the Σ-protocols presented in this section.

Corollary 1. *For $A = \mathbb{Z}$, $\Omega_n = \{0,1\}^n$, and ω as in Sect.* 3.2, *the Σ-Protocols* 1, 1.5, 2, 3, *and* 4 *have error probability* 2^{-n} *and communication complexity linear in n.*

Proof. The error probability follows immediately from $|\Omega_n| = 2^n$. The communication of Protocol 1 consists of four vectors of length $m = 2n - 1$ and n bits. The Protocols 1.5, 2, and 4 consist of up to four instances of Protocol 1. Protocol 3 in addition runs Protocol 0 n times, which has constant communication complexity. □

4. Examples

4.1. *Quadratic Residuosity*

Let N be a composite number, and let y be a nonsquare mod N. Then we can set $R = GF(2)$, $S = X = \mathbb{Z}_N^*$, $f(r,s) = y^r s^2 \bmod N$, $A = GF(2)$.

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 nonzero 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 $\mathbb{Z}_N^*$. 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 the following:

Corollary 2. *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* $\mathbb{Z}_N^*$ *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 nonsquare 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 nonsquare modulo N, and this improves the complexity of the classical protocol for this problem from [8] by a factor of n. Again, one can verify that we do not need a nonsquare 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. *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 that q divides $\phi(N)$. The plaintext space for the encryption would be $R = \mathbb{Z}_q$, 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 $A = \mathbb{Z}_q$ and $\Omega_n = \mathbb{Z}_q^n$ gives a proof of knowledge 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 [9], a different type of encryption function is proposed, also based on a composite modulus N and two elements $g, h \in \mathbb{Z}_N^*$. The encryption function is $f(m, s) = g^m h^s \bmod N$. Here m is the message chosen in $\mathbb{Z}_M$ 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 $\mathbb{Z}_N^*$.

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 $R = \mathbb{Z}_M$, $S = \mathbb{Z}$, $X = \mathbb{Z}_N^*$, and $A = \mathbb{Z}$. 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 nonzero plaintext.

4.3. *Discrete Log in a Group of Unknown Order*

In Sect. 6, we will show how to prove knowledge of a discrete log in $\mathbb{Z}_N^*$ with a modified version of our framework.

5. Zero-Knowledge Proof of Knowledge

To have small soundness error probability in our protocols, it is desirable that the challenge space Ω_n is exponentially big. Then, however, Protocol 1 cannot be proven to be zero-knowledge because a deviating verifier could make its challenge depend on the first message **a** by P, and then the standard simulation by rewinding fails. A standard solution to this problem is to let V commit to his challenge in advance. Our protocol uses the following problem-dependent commitment scheme: From a Σ-protocol for $\mathcal{R} = \{(x, w)\}$ one can construct a commitment scheme with public key x as follows: Let e be a possible challenge and (a, e, z) a transcript generated by the HVZK simulator. Then, a is a commitment to e with opening information z. To verify, one has to validate

P Input: $(\mathbf{w}, \mathbf{s}) \in R^n \times S^n$		V Input: $\mathbf{x} \in X^n$	
$(\mathbf{w_P}, \mathbf{s_P}) \in_R R^n \times S^n$ $\mathbf{x_P} := f(\mathbf{w_P}, \mathbf{s_P})$			
	$\xrightarrow{\mathbf{x_P}}$		(1)
		$e \in_R \Omega_n$ $(\mathbf{d_R}, \mathbf{d_S}) \in_R R^m \times S^m$ $\mathbf{c} := \mathbf{x_P}^E \cdot f(\mathbf{d_R}, \mathbf{d_S})$	
	$\xleftarrow{\mathbf{c}}$		(2)
$(\mathbf{r}, \mathbf{u}) \in_R R^m \times S^m$ $\mathbf{a} := f(\mathbf{r}, \mathbf{u})$			
	$\xrightarrow{\mathbf{a}}$		(3)
	$\xleftarrow{e, \mathbf{d_R}, \mathbf{d_S}}$		(4)
$\mathbf{x_P}^E \cdot f(\mathbf{d_R}, \mathbf{d_S}) \overset{?}{=} \mathbf{c}$ $\mathbf{z} := \mathbf{r} + E\mathbf{w}$ $\mathbf{v} := \mathbf{u} \cdot \gamma(E, \mathbf{w}, \mathbf{s}) \cdot \delta(\mathbf{r}, E\mathbf{w})$			
	$\xrightarrow{\mathbf{z}, \mathbf{v}, \mathbf{w_P}, \mathbf{s_P}}$		(5)
		$f(\mathbf{z}, \mathbf{v}) \overset{?}{=} \mathbf{x}^E \cdot \mathbf{a}$ $f(\mathbf{w_P}, \mathbf{s_P}) \overset{?}{=} \mathbf{x_P}$	

Fig. 1. Zero-knowledge proof of knowledge of a vector of preimages of f.

(a, e, z). The commitment is perfectly hiding because (a, e, z) is distributed as the transcript of an execution where a is chosen independent of e. The computationally binding property follows from the fact that for two different openings (e, z), (e', z') with $e \neq e'$, one can compute w such that $(x, w) \in \mathcal{R}$, and hence, breaking the commitment scheme is at least as hard as breaking the underlying relation.

Figure 1 shows a 5-round protocol, which combines Protocol 1 with the technique by Bellare et al. [2]. $\overset{?}{=}$ denotes that a party checks equality and aborts if the check fails. Furthermore, it is assumed implicitly that both parties check incoming messages for membership of the respective sets and that they abort in case of failure.

P starts by generating a new random instance $\mathbf{x_P}$ of the problem and sending it to V (1), who commits to the challenge e using $\mathbf{x_P}$ (2). P then sends the first message of Protocol 1 (3), whereupon V opens the commitment to his challenge (4). P checks the opening, answers the challenge as in Protocol 1, and also sends the witness of the instance generated in the first step (5). Finally, V checks the answer and either accepts or rejects.

Informally, zero-knowledge is achieved by V committing to the challenge e with $\mathbf{c}$. In the zero-knowledge simulator, we sometimes set $\mathbf{x_P} := \mathbf{x} \cdot f(\mathbf{w_P}, \mathbf{s_P})$. From a preimage of $\mathbf{x_P}$, $(\mathbf{w}, \mathbf{s})$ with $f(\mathbf{w}, \mathbf{s}) = \mathbf{x}$ can then be computed in polynomial time, which trivially allows us to simulate the conversion.

In the following, we will formally prove that the protocol is indeed is a zero-knowledge proof of knowledge as defined by the following definitions.

Definition 3 (Proof of knowledge). Let $\mathcal{R}$ be a relation, and x common input to P and V. A proof of knowledge has the following properties:

- Completeness: If $(x, w) \in \mathcal{R}$ and w is input to P, V always accepts.
- Knowledge soundness: Let P^* be any prover, and $\epsilon(x)$ the probability that V accepts with common input x. Then there exists a knowledge extractor M which efficiently computes w from x such that $(x, w) \in \mathcal{R}$ with at most $\frac{|x|^c}{\epsilon(x)-\kappa(x)}$ rewindable black-box accesses to P^*. κ is called the knowledge error.

Definition 4 (Zero-knowledge). A proof is zero-knowledge if for any verifier V^*, there exists an expected polynomial-time simulator M_{V*} that takes input x and generates a transcript distributed the same way as a transcript of P and V^*, having rewindable black-box access to V^*.

5.1. *Zero-Knowledge*

Zero-knowledge proofs require the construction of a simulator M_{V*} that generates a protocol run using rewindable black-box access to a possibly cheating verifier V^*. The resulting transcript must be indistinguishable to a normal protocol execution in some way. Our protocol is perfect zero-knowledge. The proof is inspired by Bellare et al. [2].

Test Algorithm The basic building block of the simulator is a so-called test algorithm. Its idea consists in running the last three steps (the actual Σ-protocol) first with some random $\mathbf{a}$, as an honest prover would, and then with an $\mathbf{a}$ prepared for the challenge we got from V^*. If V^* sends the same challenge again, we can generate an accepting transcript. If that is not true, i.e., V^* is able to open his commitment in two different ways, we are able to extract a witness of $\mathbf{x_P}$. The test algorithm works as follows:

1. Take $\mathbf{x_P}$ as input and send it to V^*.
2. Wait for $\mathbf{c}$ from V^*.
3. Generate $\mathbf{a}'$ as an honest prover would and send it to V^*.
4. Wait for $E', \mathbf{d}'_\mathbf{R}, \mathbf{d}'_\mathbf{S}$ from V^*, and check whether $(\mathbf{x_P})^{E'} \cdot f(\mathbf{d}'_\mathbf{R}, \mathbf{d}'_\mathbf{S}) = \mathbf{c}$. If false, return `fail`.
5. Rewind V^* to the state before step 3, sample $(\mathbf{z}, \mathbf{v}) \in_R R^m \times S^m$, compute $\mathbf{a} := f(\mathbf{z}, \mathbf{v}) \cdot (\mathbf{x}^E)^{-1}$, and send it to V^*.
6. Wait for $E, \mathbf{d_R}, \mathbf{d_S}$ from V^*. If $(\mathbf{x_P})^E \cdot f(\mathbf{d_R}, \mathbf{d_S}) \neq \mathbf{c}$, repeat from step 5.
7. Return $\mathbf{c}, E', \mathbf{d}'_\mathbf{R}, \mathbf{d}'_\mathbf{S}, E, \mathbf{d_R}, \mathbf{d_S}, \mathbf{z}, \mathbf{v}$.

It is straightforward to see that the algorithm terminates in expected polynomial time: Let g be the number of messages $\mathbf{a}$ that are answered correctly by V^*. Since $\mathbf{a} \in \mathrm{Im}(f)^m$, $\frac{g}{|\mathrm{Im}(f)|^m}$ is the probability that the algorithm enters the loop, and $\frac{g-1}{|\mathrm{Im}(f)|^m}$ the one that the loop is left. If $g > 1$, the expected running time is $\frac{g}{|\mathrm{Im}(f)|^m} \cdot \frac{|\mathrm{Im}(f)|^m}{g-1} \in O(1)$. To handle the case of $g = 1$, we try to find the witness w for x parallel to the loop. Because the probability of guessing a witness is $(\frac{1}{|\mathrm{Im}(f)|})^n$, and $n \leq m$, the algorithm then terminates in expected time $O(1)$ for any g.

Witness Extraction A further algorithm used by the simulator is an algorithm extracting a witness of $\mathbf{x_P}$ if V^* can open his commitments in two different ways. We use this to extract a witness of $\mathbf{x}$ because we can generate $\mathbf{x_P}$ as a rerandomization of $\mathbf{x}$.

Witness extraction is based on the special soundness property of the Σ-protocol. Given $E, \mathbf{d_R}, \mathbf{d_S}$ and $E', \mathbf{d'_R}, \mathbf{d'_S}$ with $(\mathbf{x_P})^E \cdot f(\mathbf{d_R}, \mathbf{d_S}) = \mathbf{c} = (\mathbf{x_P})^{E'} \cdot f(\mathbf{d'_R}, \mathbf{d'_S})$, we can find $(\tilde{\mathbf{w}}, \tilde{\mathbf{s}})$ with $f(\tilde{\mathbf{w}}, \tilde{\mathbf{s}}) = \mathbf{x_P}$ similarly to the proof of Lemma 3. Thus, we can compute a witness for $\mathbf{x_P}$.

The Simulator Now we put the algorithms together. The simulator M_{V^*} works as follows:

1. Fix the randomness of V^*.
2. Call the test algorithm with $\mathbf{x_P} = \mathbf{x} \cdot f(\tilde{\mathbf{w}}_\mathbf{P}, \tilde{\mathbf{s}}_\mathbf{P})$ for $(\tilde{\mathbf{w}}_\mathbf{P}, \tilde{\mathbf{s}}_\mathbf{P}) \in_R R^n \times S^n$.

   - If it returns `fail`, return `fail` as well.
   - If it returns two different challenges $E \neq E'$, a preimage of $\mathbf{x}$ can be computed with the witness extraction algorithm because $(\mathbf{x_P})^E \cdot f(\mathbf{d_R}, \mathbf{d_S}) = (\mathbf{x_P})^{E'} \cdot f(\mathbf{d'_R}, \mathbf{d'_S})$ and $\mathbf{x_P} = \mathbf{x} \cdot f(\tilde{\mathbf{w}}_P, \tilde{\mathbf{s}}_P)$. After computing $(\mathbf{w}, \mathbf{s})$ with $\mathbf{x} = f(\mathbf{w}, \mathbf{s})$, repeat the following:
     (a) Generate $\mathbf{x_P} = f(\mathbf{w_P}, \mathbf{s_P})$ with $(\mathbf{w_P}, \mathbf{s_P}) \in_R R^n \times S^n$.
     (b) Rewind V^* to the beginning.
     (c) Call the test algorithm with $\mathbf{x_P}$. If it returns two different challenges, output $\mathbf{x_P}, \mathbf{c}, \mathbf{a}, (E, \mathbf{d_R}, \mathbf{d_S}), (\mathbf{z}, \mathbf{v}, \mathbf{w_P}, \mathbf{s_P})$ and return. $(\mathbf{z}, \mathbf{v})$ can be computed using $(\mathbf{w}, \mathbf{s})$. Otherwise, continue with the loop.
   - If it returns two times the same challenge $E = E'$, repeat the following:
     (a) Generate $\mathbf{x_P} = f(\mathbf{w_P}, \mathbf{s_P})$ with $(\mathbf{w_P}, \mathbf{s_P}) \in_R R^n \times S^n$.
     (b) Rewind V^* to the beginning.
     (c) Call the test algorithm with $\mathbf{x_P}$. If it returns two equal challenges, output $\mathbf{x_P}, \mathbf{c}, \mathbf{a}, (E, \mathbf{d_R}, \mathbf{d_S}), (\mathbf{z}, \mathbf{v}, \mathbf{w_P}, \mathbf{s_P})$ and return. Otherwise, continue with the loop.

M_{V^*} terminates in expected polynomial time because it only calls algorithms with expected polynomial running time and because the loops stop with the same probability as they were entered.

The proof that the distribution of the output of the simulator complies with the one of a normal execution is, mutatis mutandis, the same as the proof given by Bellare et al. [2]. A central element is that one loops runs until V^* returns two different challenges, and the other loop runs until V^* return twice the same challenge. In the second case, this is needed to generate a correct answer. In the first case however, the simulator knows a witness for $\mathbf{x}$ and could answer every challenge correctly. The reason to wait for two different challenges is that some challenges are less likely to be generated by the second loop than by a normal execution. For example, a cheating verifier could open the commitments to some challenge E only for one choice of $\mathbf{a}$. Such a challenge would never be output by the second loop because it requires two times the same challenge for two different $\mathbf{a}$. This is compensated by the first loop, which outputs such challenges with higher probability than in a normal execution.

5.2. *Knowledge Extraction*

In this section, we show how to extract a witness for $\mathbf{x}$ from any P^* that has success probability bigger than $2/|\Omega_n|$. Bellare and Goldreich [1] proved that deterministic provers in proofs of knowledge are as strong as probabilistic provers; therefore, it suffices to present a knowledge extractor for a deterministic prover. Our algorithm is based on the knowledge extraction with a plain Σ-protocol, which uses the special soundness property of Σ-protocols. The goal is to get the answer $(\mathbf{z}, \mathbf{v})$ for two different challenges e but the same previous message $\mathbf{a}$. The extractor consists of three phases: Generating an accepting transcript to learn a preimage of $\mathbf{x_P}$, generating a transcript with equivocable commitments, and generating a transcript with the same $\mathbf{a}$ as previously, but a different challenge.

The standard proof that a Σ-protocol is a proof of knowledge does not translate directly to the extended protocol. This is because openings of the commitments in phases 2 and 3 have to be independent of $(\mathbf{w_P}, \mathbf{s_P})$ sent by P^* in the last message of phase 1. Otherwise, P^* could behave differently when receiving the opening $(\mathbf{d_R}, \mathbf{d_S})$, which breaks the proof because it relies on that fact P^*'s success probability does not change. Our framework for self-reducible problems does not guarantee the required independence. Therefore, even though we can open the generated commitment in both ways after learning $(\mathbf{w_P}, \mathbf{s_P})$, we need to generate a new random commitment after that, which necessitates a three-phase knowledge extractor.

Equivocable Commitments An essential part of the proof is the notion of equivocable commitments, i.e., commitments that can be opened to any $e \in \Omega_n$. Such commitments can be generated as follows: We choose $(\mathbf{u_R}, \mathbf{u_S}) \in_R R^m \times S^m$ and compute $\mathbf{c} := f(\mathbf{u_R}, \mathbf{u_S})$. To open, let

$$\begin{aligned} \mathbf{d_R} &:= \mathbf{u_R} - E\mathbf{w_P}, \\ \mathbf{d_S} &:= \mathbf{u_R} \cdot \gamma(E, \mathbf{w_P}, \mathbf{s_P})^{-1} \cdot \delta(E\mathbf{w_P}, \mathbf{u_R} - E\mathbf{w_P})^{-1}. \end{aligned} \tag{15}$$

Then, it is easy to see that

$$\begin{aligned} \mathbf{x_P}^E \cdot f(\mathbf{d_S}, \mathbf{d_R}) &= f\big(E\mathbf{w_P}, \gamma(E, \mathbf{w_P}, \mathbf{s_P})^{-1}\big) \\ &\quad \cdot f\big(\mathbf{u_R} - E\mathbf{w_P}, \mathbf{u_R} \cdot \gamma(E, \mathbf{w_P}, \mathbf{s_P})^{-1} \cdot \delta(E\mathbf{w_P}, \mathbf{u_R} - E\mathbf{w_P})^{-1}\big) \\ &= f(\mathbf{u_R}, \mathbf{u_S}) = \mathbf{c}. \end{aligned}$$

We now give a description of the three phases of the knowledge extractor. First, we need to learn a witness $(\mathbf{w_P}, \mathbf{s_P})$ for $\mathbf{x_P}$ in order to be able to generate equivocable commitments. We just execute the protocol as a regular verifier and restart whenever P^* fails to send a correct message. If the execution is successful, the extractor learns $(\mathbf{w_P}, \mathbf{s_P})$ at the end. Overall, this takes time inverse to the success probability of P^*, which complies with the definition of a proof of knowledge.

In the second phase, we try to generate a new transcript with equivocable commitments. To do so, we rewind P^* to the state after step 1 (see the protocol on p. 301), choose new randomness for V, and run the protocol while generating an equivocable commitment $\mathbf{c}$ using $(\mathbf{w_P}, \mathbf{s_P})$ learned in the previous step. Then, we send them to P^*, wait for $\mathbf{a}$, and open our commitments to a randomly chosen challenge.

To generate a transcript with a different challenge, we rewind P^* to the state after step 3, choose a new challenge e' uniformly at random, and open the commitment to it as described above. We then send the new challenge and the opening information to P^*, and wait for an answer. It is crucial that the opening in the second and third phases is independent of $(\mathbf{w_P}, \mathbf{s_P})$ sent by P^* in the first phase because $(\mathbf{w_P}, \mathbf{s_P})$ could be dependent of the opening there. We achieve independence by generating new commitments.

Another component that we use in extractor is a *random experiment* returning 0 or 1, where the probability of returning 1 is the same as the success probability of P^*. This is straightforward, we just try to generate an accepting transcript with an independent instance of P^* using fresh randomness. If we are successful, we return 1, and 0 otherwise.

Putting the pieces together, the extractor works as follows:

1. Loop:
   - Try to generate an accepting transcript following V's algorithm to learn $(\mathbf{w_P}, \mathbf{s_P})$. If successful, go to phase 2.
2. Loop:
   - Rewind P^* to the state after step 1 and try to generate a transcript with equivocable commitments. If successful, go to phase 3.
3. Loop:
   - Rewind P^* to the state after step 3 and try to generate a transcript with a fresh random challenge. If successful, extract the witness $(\mathbf{w}, \mathbf{s})$ and stop.
   - Do the random experiment once and go to phase 2 if it has been successful 8 times.

Theorem 2. *The above algorithm is a valid knowledge extractor for deterministic provers with knowledge error* $2/|\Omega_n|$.

Proof. An essential part of the proof is the notion of randomness of V, which is not learned by P^*. This is some of the randomness used to generate equivocable commitments, namely the choice of $(\mathbf{d_R}, \mathbf{d_S})$ in $f^{-1}(\mathbf{c} \cdot (\mathbf{x_P}^{\omega(e)})^{-1})$. In other words, this is the choice in all possible openings of $\mathbf{c}$ to e. Note that $\mathbf{x_P}$ is fixed for a deterministic prover.

Let $C := \mathrm{Im}(f)$ be the set of all commitments, and $D_{\mathbf{c},e} := f^{-1}(\mathbf{c} \cdot (\mathbf{x_P}^{\omega(e)})^{-1})$ the set of all possible openings $(\mathbf{d_R}, \mathbf{d_S})$ for $\mathbf{c} \in C$ and $e \in \Omega_n$. Note that $D_{c,e} \neq \emptyset$ for all $c, e \in C \times \Omega_n$ because $C = \mathrm{Im}(f)$ is a group by (1) and (2), and thus, it is possible to open a commitment to any e. The random choice $(\mathbf{u_R}, \mathbf{u_S})$ of the extractor generating the commitment in phase 2 determines a $d \in D_{c,e}$ for every $c \in C, e \in \Omega_n$. We denote this determination by $q : C \times \Omega_n \to \bigcup_{c \in C, e \in \Omega_n} D_{c,e}$, and we denote by Q the set of all q induced by all $(\mathbf{u_R}, \mathbf{u_S}) \in R^m \times S^m$. By definition, $q(c,e) \in D_{c,e}$ for all $q \in Q$, $c \in C, e \in \Omega_n$. From (15) it easily follows that every $(\mathbf{u_R}, \mathbf{u_S}) \in R^m \times S^m$ induces a different $q \in Q$ and that for a uniformly at random chosen $(\mathbf{u_R}, \mathbf{u_S}) \in R^m \times S^m$, $q(c,e)$ is uniformly distributed in $D_{c,e}$ for all $c \in C, e \in \Omega_n$. Therefore, $q(c,e)$ is uniformly distributed in $D_{c,e}$ for a uniformly chosen $q \in Q$ for all $c \in C, e \in \Omega_n$. Finally, we denote by K all successful communications, i.e., $K := \{(c,e,d) \mid c \in C, e \in \Omega_n, d \in D_{c,e}\}$.

There exists a subset $G \subset K$ of all successful protocol executions with V^*. In the second loop, we search for an entry $(c, e, q(c, e)) \in G$ in the second loop for a uniformly chosen $q \in Q$. Recall that we cannot use the commitment from the first loop because we cannot open the commitment to an arbitrary challenge. Finally, we fix c and try to find an entry $(c, e', q(c, e')) \in G$ with $e' \neq e$, which allows us to compute the witness. Since a uniformly distributed $q \in Q$ implies that $q(c, e)$ and $q(c, e')$ are uniformly distributed and independent, we can assume for simplicity that the extractor uniformly chooses (c, e, d) in the second phase and (c, e', d') in the third phase instead.

Now, we prove that the extractor is able to extract a witness in expected time $O(\frac{1}{\epsilon - 2/|\Omega_n|})$, where ϵ denotes the success probability of P^*.

We call $(q, c) \in Q \times C$ heavy if $\{(c, e, q(c, e)) \mid e \in \Omega_n\}$ contains a fraction of at least $\epsilon/2$ entries in G, i.e., at least 2 entries because $\epsilon > 2/|\Omega_n|$. Half of the entries in K have a heavy (q, c). Otherwise, let H the set of all nonheavy $(q, c) \in Q \times C$, and $K' := \{(c, e, q(c, e)) \mid (q, c) \in H, e \in \Omega_n\} \subset K$. Then,

$$|K| = \frac{|G|}{\epsilon} \overset{\heartsuit}{<} \frac{2|K' \cap G|}{\epsilon} \overset{\text{🍃}}{<} \frac{2|K'|\epsilon/2}{\epsilon} = \left|S'\right| \leq |K|,$$

which is a contradiction. It follows from the definition of G, the contradiction assumption ($\heartsuit$), and the definition of K' (🍃).

The expected running time of the first and second loops is $T_1 = T_2 = 1/\epsilon$ because ϵ is the success probability of P^*. By the random experiment, the expected maximal running time of the third loop is $T_3 = 8/\epsilon$, respectively. And by the union bound, the probability that the random experiment fails at least $T_3/2$ times is $1/2$.

Since half of the entries in G have a heavy (q, c), the probability of fixing such an (q, c) in the first loop is $1/2$. In this case, the expected time to find a successful entry $(c, e, d) \in G$ is $2/\epsilon$ because their share is at least $\epsilon/2$. By Markov's inequality, the probability of finding a successful entry in less than $T_3/2$ tries is at least $1 - \frac{2/\epsilon}{T_3/2} = 1 - \frac{4\epsilon}{8\epsilon} = \frac{1}{2}$. Therefore, with probability $1/2 \cdot 1/2 = 1/4$, we are successful in the third loop.

We conclude that with at least constant probability $1/2 \cdot 1/4 = 1/8$, we can extract a witness in time $T_1 + T_2 + T_3 = 10/\epsilon$. Thus, the extractor requires expected time $\frac{8 \cdot 10}{\epsilon} \in O(\frac{1}{\epsilon - 2/|\Omega_n|})$. □

6. Discrete Log in a Group of Unknown Order

Let N be an arbitrary k-bit number, and $g \in \mathbb{Z}_N^*$. Then we will set $R = \mathbb{Z}$, S to be the trivial group with one element, $X = \mathbb{Z}_N^*$, and $f(r, 1) = g^r \bmod N$. We also set $A = \mathbb{Z}$. This does not quite satisfy our framework since R is not finite, but we will fix this shortly. Throughout this section, we will ignore the second input to f since it is constantly 1.

The construction behind Theorem 1 implies that we can satisfy the conditions in our framework by constructing the set Ω_n as the subset of $\mathbb{Z}^n$ consisting of binary strings. Recall that the construction defines $m := 2n - 1$ and $\omega : \mathbb{Z}^n \to \mathbb{Z}^{m \times n}$ as follows: Let the ith column of $\omega(e)$ consist of $i - 1$ zeros, followed by e and $n - i$ zeros.

In this case, Protocol 1 has to be tweaked slightly: instead of choosing $\mathbf{r}$ uniformly in R^n, which does not make sense for $R = \mathbb{Z}$, we choose the entries as uniform $(2\log n + 2k)$-bit numbers. This choice ensures both that $f(\mathbf{r})$ will be statistically close to uniform in $\operatorname{Im}(f)^m$ with respect to k and that the entries in $\mathbf{z}$ will be statistically close to uniform $(2\log n + 2k)$-bit numbers. This follows from the fact that the entries in $E\mathbf{w}$ will be at most n times bigger than the ones of $\mathbf{w}$ because E is an $m \times n$-matrix with all entries in $\{0, 1\}$. Thus, the entries of $E\mathbf{w}$ are at most $n2^k$ if the entries of $\mathbf{w}$ are smaller than N.

The protocol now becomes an interactive proof that the input numbers $x_1, \dots, x_n$ 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.

To prove the properties of the protocol more formally, we need the following three lemmas. For readability, we will write $\log n$ instead of $\lceil \log n \rceil$; recall that $n \le 2^{\lceil \log n \rceil}$.

Lemma 7. *Let $n, l, N \in \mathbb{N}$, $l \ge 2k + \log n$, and $N \le 2^k$. If $\mathbf{r}$ is a vector of $m = 2n - 1$ uniformly and independently chosen l-bit numbers, then $f(\mathbf{r})$ is distributed statistically close to the uniform distribution in $(\mathbb{Z}_N^*)^m$, with respect to k.*

Proof. First, we determine the statistical distance for one entry. Let φ denote Euler's totient function, that is, $\varphi(N) = |\mathbb{Z}_N^*|$. Since

$$f(r) = g^r = g^{r \bmod \varphi(N)} = f\bigl(r \bmod \varphi(N)\bigr),$$

it is easy to see that the probability to sample x is

$$\Pr\bigl[f(r) = x\bigr] \in \left\{ \left\lceil \frac{2^l}{\varphi(N)} \right\rceil \cdot \frac{1}{2^l}, \left\lfloor \frac{2^l}{\varphi(N)} \right\rfloor \cdot \frac{1}{2^l} \right\}$$

for all $x \in \mathbb{Z}_N^*$ because r is chosen uniformly at random in $[0, 2^l - 1]$. It follows that

$$\Pr\bigl[f(r) = x\bigr] \le \left(\frac{2^l}{\varphi(N)} + 1 \right) \cdot \frac{1}{2^l} = \frac{1}{\varphi(N)} + \frac{1}{2^l}$$

for all $x \in \mathbb{Z}_N^*$, and, similarly,

$$\Pr\bigl[f(r) = x\bigr] \ge \frac{1}{\varphi(N)} - \frac{1}{2^l}$$

for all $x \in \mathbb{Z}_N^*$. It follows that the statistical distance of one entry is

$$\begin{aligned} \sum_{x \in \mathbb{Z}_N^*} \left| \Pr\bigl[f(r) = x\bigr] - \frac{1}{\varphi(N)} \right| &\le \sum_{x \in \mathbb{Z}_N^*} \frac{1}{2^l} = \frac{\varphi(N)}{2^l} \\ &\le \frac{N}{2^{2k + \log n}} \le 2^{-k - \log n}. \end{aligned}$$

The latter two inequalities follow from $\varphi(N) \le N \le 2^k$. Finally, the statistical distance between $f(\mathbf{r})$ and a uniformly chosen vector is less than

$$m \cdot 2^{-k - \log n} \le 2n \cdot 2^{-k - \log n} \le 2^{-k+1},$$

which is negligible in k. □

Lemma 8. *Let* $\mathbf{y} \in (\mathbb{Z}_N^*)^n$ *be chosen according to a distribution statistically close to the uniform distribution, and* $\mathbf{x}$ *an arbitrary element of* $(\mathbb{Z}_N^*)^n$*. Then,* $\mathbf{x} \cdot \mathbf{y}$ *is also distributed statistically close to the uniform distribution.*

Proof. That the statistical distance of $\mathbf{x} \cdot \mathbf{y}$ to the uniform distribution is

$$\sum_{\mathbf{z} \in (\mathbb{Z}_N^*)^n} \left| \Pr[\mathbf{x} \cdot \mathbf{y} = \mathbf{z}] - \frac{1}{|\mathbb{Z}_N^*|^n} \right| = \sum_{\mathbf{z} \in (\mathbb{Z}_N^*)^n} \left| \Pr\left[\mathbf{y} = \mathbf{x}^{-1} \cdot \mathbf{z}\right] - \frac{1}{|\mathbb{Z}_N^*|^n} \right|$$
$$= \sum_{\mathbf{z}' \in (\mathbb{Z}_N^*)^n} \left| \Pr\left[\mathbf{y} = \mathbf{z}'\right] - \frac{1}{|\mathbb{Z}_N^*|^n} \right|$$

because multiplication with a fixed element is bijective on a group. The lemma follows directly. □

Lemma 9. *Let* $n, m, k, l, l' \in \mathbb{N}$ *be such that* $l' \geq k + l + 2\log n$ *and* $m = 2n - 1$, $\mathbf{a}$ *be chosen arbitrarily in* $[-2^l n, 2^l n]^m$*, and* $\mathbf{b}$ *be chosen uniformly in* $[0, 2^{l'} - 1]^m$*. Then,* $\mathbf{a} + \mathbf{b}$ *is distributed statistically close to the uniform distribution on* $[0, 2^{l'} - 1]^m$*, with respect to* k*.*

Proof. Let

$$d(c) := \begin{cases} 2^{-l'}, & c \in [0, 2^{l'} - 1], \\ 0 & \text{otherwise} \end{cases}$$

be the density function of the uniform distribution on l'-bit numbers. For one entry $a + b$ in the vector $\mathbf{a} + \mathbf{b}$, the statistical distance is

$$\sum_{c \in [-2^l n, 2^{l'} + 2^l n - 1]} \left| \Pr[a + b = c] - d(c) \right|.$$

For any c,

$$\Pr[a + b = c] = \sum_{i = -2^l n}^{2^l n} \Pr[a = i] \Pr[b = c + i].$$

It follows that for $c \in [2^l n, 2^{l'} - 2^l n - 1]$,

$$\left| \Pr[a + b = c] - d(c) \right| = \left| \sum_{i = -2^l n}^{2^l n} \Pr[a = i] \cdot 2^{-l'} - 2^{-l'} \right| = 0,$$

and for $c < 2^l n$,

$$\left| \Pr[a + b = c] - d(c) \right| = \left| \sum_{i = -c}^{2^l n} \Pr[a = i] \cdot 2^{-l'} - d(c) \right| \leq 2^{-l'},$$

since $d(c) \in \{0, 2^{-l'}\}$. The same holds by symmetry for $c > 2^{-l'} - 2^l n - 1$. Summing up, the statistical distance is less than

$$\sum_{c\in[-2^l n, 2^l n-1]\cup[2^{-l'}-2^l n, 2^{l'}+2^l n-1]} 2^{-l'} = 4 \cdot 2^l n \cdot 2^{-l'} \leq 2^{2-k-\log n}.$$

Therefore, the statistical distance of the whole vector is less than

$$m \cdot 2^{2-k-\log n} \leq 2n \cdot 2^{2-k-\log n} \leq 2^{3-k},$$

which is negligible in k. □

Lemma 10. *Protocol* 1 *instantiated for the discrete log in* $\mathbb{Z}_N^*$ *case is a honest-verifier statistical zero-knowledge* Σ*-protocol proving that the input numbers* $x_1, \ldots, x_n$ *are all in the group generated by* g *and that the prover knows the discrete logarithms. Let* k *be the bit length of* N*. Then the communication complexity is* $O(nk + n \log n)$ *bits, the error probability* 2^{-n}*, and the computational complexity is* $O(n^2)$ *multiplications modulo* N*.*

Proof. Completeness is trivial by construction, and soundness follows exactly as in the proof of Lemma 3. The honest-verifier zero-knowledge simulator works similarly as well: Choose $e \in_R \Omega_n$, the entries of $\mathbf{z}$ as uniform $(2 \log n + 2k)$-bit numbers, and let $\mathbf{a} := f(\mathbf{z}) \cdot (\mathbf{x}^{\omega(e)})^{-1}$. $\mathbf{a}$ is then distributed statistically close to uniform in $\mathrm{Im}(f)^m$ by Lemma 8 because $f(\mathbf{z})$ is so by Lemma 7. In a real execution, $\mathbf{a}$ is also statistically close to uniform in $\mathrm{Im}(f)^m$ by Lemma 7, e has the same distribution, and the entries of $\mathbf{z}$ are statistically close to uniform $(2 \log n + 2k)$-bit numbers by Lemma 9 with $l = k$. It follows that the protocol is honest-verifier statistical zero-knowledge.

The communication complexity is dominated by $\mathbf{z}$, which is a vector of n numbers with bit length $O(\log n + k)$. Furthermore, the verifier has to compute $\mathbf{x}^E \cdot \mathbf{a}$, which consists of $O(n^2)$ 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. Therefore, if we choose, e.g., $n = k$, our solution saves a factor k in both the communication and computational complexity.

6.1. *Zero-Knowledge Proof of Knowledge*

The protocol in Fig. 1 can be adapted to $R = \mathbb{Z}$ similarly. Let the entries of $\mathbf{w_P}$ and $\mathbf{d_R}$ be chosen as uniform $(\log n + 2k)$-bit and $(3 \log n + 3k)$-bit numbers, respectively. Then, both $f(\mathbf{w_P})$ and $f(\mathbf{d_R})$ are statistically close to uniform in $\mathrm{Im}(f)^n$ and $\mathrm{Im}(f)^m$, respectively, by Lemma 7. Furthermore, the prover chooses the entries of $\mathbf{r}$ as uniform $(2 \log n + 2k)$-numbers as above.

The zero-knowledge simulator in Sect. 5.1 can be modified as follows: The test algorithm samples $\mathbf{a}'$ like an honest prover and the entries of $\mathbf{z}$ as uniform $(2 \log n + 2k)$-bit numbers. As in the proof of Lemma 10, $\mathbf{a} = f(\mathbf{z}) \cdot (x^{\omega(e)})^{-1}$ is then statistically close to the uniform distribution, and $\mathbf{z}$ is statistically close to a vector of uniform $(2\log n + 2k)$-bit numbers. In the simulator algorithm, the entries of both $\tilde{\mathbf{w}}_\mathbf{P}$ and $\mathbf{w}_\mathbf{P}$ can be chosen as uniform $(\log n + 2k)$-bit numbers. If so, $\mathbf{x} \cdot f(\tilde{\mathbf{w}}_\mathbf{P})$ and $f(\mathbf{w}_\mathbf{P})$ are statistically close to uniform by Lemma 7 and 8. It follows that all inputs to V^* are statistically close to a real execution, and hence, statistical zero-knowledge of the modified protocol can be proven in the same way as perfect zero-knowledge of the unmodified one.

For knowledge extraction, we generate equivocable commitments by choosing the entries of $\mathbf{u}_\mathbf{R}$ as uniform $(3\log n + 3k)$-bit numbers. Thus, $\mathbf{d}_\mathbf{R} := \mathbf{u}_\mathbf{R} - E\mathbf{w}_\mathbf{P}$ is statistically close to a vector of uniform $(3\log n + 3k)$-bit numbers by Lemma 9 with $l = \log n + 2k$ because the entries $E\mathbf{w}_\mathbf{P}$ are at most $(2\log n + 2k)$-bit numbers similarly as for $E\mathbf{w}$ above. By construction of the knowledge extractor in Sect. 5.2, the statistical distance of any message sent to P^* is at most a negligible $\mu(k)$ compared to a real execution. Therefore, the knowledge extractor finishes in time

$$O\left(\frac{1}{\epsilon - \mu(k)}\right) \subset O\left(\frac{1}{\epsilon - \kappa}\right)$$

if $\kappa \geq \mu(k)$. We conclude that the adapted protocol is a proof of knowledge with knowledge error $\kappa := \max(2/|\Omega_n|, \mu(k))$.

6.2. *Interval Proof*

The protocol in this section does not require that the prover knows the group order of $X = \mathbb{Z}_N^*$. If we assume that the group order is indeed unknown, a similar protocol proves in addition a limit on $\mathbf{w}$. Let $\mathbf{w} \in [0, u]^n$ for some $u \in \mathbb{N}$, $l := \log n + k + \log u$, and let P choose the entries of $\mathbf{r}$ as uniformly random l-bit numbers. Then, $\mathbf{z} = \mathbf{r} + E\mathbf{w}$ is distributed statistically close to the uniform distribution on $[0, 2^l]^m$ because the entries of $E\mathbf{w}$ are at most nu, similarly to above. On the other hand, the following lemma establishes that if the prover can answer two different challenges within $[0, 2^l]^m$, $\mathbf{w}$ is in $[-2^{l+n}, 2^{l+n}]^m \approx [-un2^{k+n}, un2^{k+n}]^m$.

Lemma 11. *Let $\mathbf{x} \in \mathbb{Z}_N^*$, $e \neq e' \in \{0,1\}^n$, and $\mathbf{z}, \mathbf{z}' \in [0, 2^l]^m$ such that $\mathbf{z}$ and $\mathbf{z}'$ are accepted as replies to the challenges e and e', respectively, with the same first message from P. Then, there exists $\mathbf{w} \in [-2^{l+n}, 2^{l+n}2]^m$ such that $g^\mathbf{w} = \mathbf{x}$.*

Proof. Let the $m \times n$ matrices $E := \omega(e)$ and $E' := \omega(e')$ be constructed from e and e' as described in Sect. 3. The ith column of E and E' consists of $i-1$ zeros, followed by some e and $e' \in \{0,1\}^n$, respectively, and $n-i$ zeros. $(E - E')$ has the same structure for $e - e' \in \{0, \pm 1\}^n$. It is easy to see that the columns of $(E - E')$ are linearly independent if $e \neq e'$. Thus, $(E - E')$ has rank n and is left-invertible.[2] We call the left inverse $(E - E')^{-1}$ and define $\mathbf{w} := (E - E')^{-1}(\mathbf{z} - \mathbf{z}')$. $\mathbf{z}, \mathbf{z}'$ are accepted replies only if $g^\mathbf{z} = \mathbf{x}^E \cdot \mathbf{a}$ and $g^{\mathbf{z}'} = \mathbf{x}^{E'} \cdot a$. It follows that $g^{\mathbf{z}-\mathbf{z}'} = \mathbf{x}^{E-E'}$. Applying $(E - E')^{-1}$ on both sides gives

$$g^\mathbf{w} = g^{(E-E')^{-1}(\mathbf{z}-\mathbf{z}')} = \mathbf{x}^{(E-E')^{-1}(E-E')} = \mathbf{x}.$$

[2] Recall that $m = 2n - 1 > n$.

We now prove that $\mathbf{w} \in [-2^{l+n}, 2^{l+n}]^m$. Let $\mathbf{y} := \mathbf{z} - \mathbf{z}'$. If $\mathbf{z}, \mathbf{z}' \in [0, 2^l]^m$, $\mathbf{y} \in [-2^l, 2^l]^m$. $\mathbf{w}$ is the solution of $(E - E')\mathbf{w} = \mathbf{z} - \mathbf{z}'$. Let j be the first position where e and e' differ. Without loss of generality, we can assume that $d := e - e'$ has a 1 in that position. $(E - E')$ then has the following structure, starting with $j - 1$ empty rows:

$$\begin{pmatrix} \vdots & & \\ 0 & & \\ 1 & \ddots & \vdots \\ d_{j+1} & \ddots & 0 \\ \vdots & \ddots & 1 \\ d_n & & d_{j+1} \\ & \ddots & \vdots \\ 0 & & d_n \end{pmatrix}$$

Therefore, the jth equation of the system $(E - E')\mathbf{w} = \mathbf{z} - \mathbf{z}'$ is $w_1 = z_j$ for w_i denoting the ith entry of $\mathbf{w}$. Thus, $|w_1| \leq 2^l$. For induction, assume that $|w_h| \leq 2^{l+h-1}$ for $h < i$. The equation induced by the ith row of $E - E'$ is

$$\sum_{h=1}^{i-1} d_{j+i-h} w_h + w_i = y_i.$$

It follows that

$$\begin{aligned} |w_i| &= |y_i - \sum_{h=1}^{i-1} d_{j+i-h} w_h| \leq |y_i| + \sum_{h=1}^{i-1} |w_h| \\ &\leq 2^l + \sum_{h=1}^{i-1} 2^{l+h-1} = 2^l \left(1 + \sum_{h=1}^{i-1} 2^{h-1}\right) \leq 2^{l+i-1}. \end{aligned}$$

The first inequality follows from the triangle inequality and the fact that $|d_i| \leq 1$ because $d_i \in \{0, \pm 1\}$. The second inequality follows from the induction assumption. By induction, we have established that $|w_i| \leq 2^{l+n}$ for all $i \leq n$, and thus, $\mathbf{w} \in [-2^{l+n}, 2^{l+n}]^m$. □

Appendix A. Random Self-reducible Problems

The proofs given in Sect. 5 apply in similar form to the protocol by Bellare et al. [2]. In this section, we will present it for a more general class of problems, namely random self-reducible problems, instead of graph isomorphism.

Definition 5 (Random self-reducible problem). Let $\mathcal{R}_n$ be a family of relations $R \subset \{0,1\}^* \times \{0,1\}^*$. We call $\mathcal{R} := \{\mathcal{R}_n\}_{n \in \mathbb{N}}$ a random self-reducible problem if there exist rerandomizing algorithms $(\rho_x, \rho_w, \rho_x^{-1}, \rho_w^{-1})$ with the following properties for all $R \in \mathcal{R}_n \in \mathcal{R}$.

$$\begin{array}{lcl}
\hline
P & & V\\
\text{Input: } (x,w)\in R & & \text{Input: } x\\
\hline
r \in_R \{0,1\}^{O(n)} & & \\
a := \rho_x(x,r) & & \\
 & \xrightarrow{\qquad\qquad a \qquad\qquad} & \\
 & & e\in\{0,1\}\\
 & \xleftarrow{\qquad\qquad e \qquad\qquad} & \\
z := \begin{cases} r & e=0\\ \rho_w(w,r) & e=1\end{cases} & & \\
 & \xrightarrow{\qquad\qquad z \qquad\qquad} & \\
 & & \begin{cases} a \overset{?}{=} \rho_x(x,z) & e=0\\ (a,z)\overset{?}{\in} R & e=1\end{cases}\\
\hline
\end{array}$$

Fig. A.1. Σ-protocol to prove knowledge of a solution of random self-reducible problem.

- $(\rho_x(x,r), \rho_w(w,r)) \in R$ for all $(x,w) \in R$ and $r \in \{0,1\}^{O(n)}$.
- $\rho_x^{-1}(\rho_x(x,r),r) = x$ and $\rho_w^{-1}(\rho_w(w,r),r) = w$ for all $(x,w) \in R$.
- For all $(x,w) \in R$, $(\rho_x(x,r), \rho_w(w,r))$ is distributed uniformly at random in R if r is so in $\{0,1\}^{O(n)}$.
- ρ_x, ρ_w, ρ_x^{-1}, and ρ_w^{-1}, as well as sampling in R and the computation of group operations in G are polynomial in n.

Throughout this section, we will assume that $R \in \mathcal{R}_n$. Figure A.1 shows the standard knowledge proof of a solution of a random self-reducible problem. Completeness is trivial, and special soundness is straightforward: If both r and $\rho_w(w,r)$ are known, one can compute $w := \rho_w^{-1}(\rho_w(w,r),r)$. Special honest-verifier zero-knowledge follows from the randomizing property of ρ and from the following construction of (a,z) from e:

$$(a,z) := \begin{cases} (\rho_x(x,r), r \in_R \{0,1\}^{O(n)}), & e = 0,\\ (a,z) \in_R R, & e = 1.\end{cases}$$

Discrete Logarithm A first example is the problem of finding discrete logarithms. Let $\mathcal{G}_n$ be a family of representation of cyclic groups whose order has bit length n, and

$$\mathcal{R}_n := \bigl\{\bigl\{(x,w) \bigm| x \in G, w \in \mathbb{Z}_{|G|}, G = \langle g\rangle, x = g^w\bigr\} \bigm| G \in \mathcal{G}_n\bigr\}.$$

Then $\mathcal{R} := \{\mathcal{R}_n\}_{n\in\mathbb{N}}$ is a random self-reducible problem with the following algorithms for $R \in \mathcal{R}_n$. The following algorithms use $r \in \{0,1\}^{O(n)}$ to generate $s \in_R \mathbb{Z}_{|G|}$. We define

$$\rho_x(x,r) := x \cdot g^s, \qquad \rho_x^{-1}(x,r) := x \cdot g^{-s},$$

$$\rho_w(w,r) := w + s, \qquad \rho_w^{-1}(w,r) := w - s.$$

It follows that

$$\rho_x^{-1}\bigl(\rho_x(x,r),r\bigr) = \rho_x^{-1}\bigl(x \cdot g^s\bigr) = x \cdot g^s \cdot g^{-s} = x$$

and

$$\rho_w^{-1}\big(\rho_w(w,r),r\big)=\rho_w^{-1}(w+s)=w+s-s=w.$$

Trivially, $\rho_x^{-1}(\cdot,r)$ and $\rho_w^{-1}(\cdot,r)$ are the inverses of $\rho_x(\cdot,r)$ and $\rho_w(\cdot,r)$, and the output of ρ_x and ρ_w is distributed uniformly at random in G and $\mathbb{Z}_{|G|}$, respectively, because $s \in_R \mathbb{Z}_{|G|}$.

Quadratic Residues Another example is the computation of quadratic residues modulo N, where $N = p \cdot q$ is an n-bit RSA modulus. Let $R := \{(x,w) \mid x,w \in \mathbb{Z}_N, x = w^2\}$. Rerandomizing multiplies w by a random $s \in \mathbb{Z}_N$ and x by s^2. It is easy to see that a properly compiled family of the given relations matches the definition of a random self-reducible problem.

Graph Isomorphism The last example is the problem of finding an isomorphism for two isomorphic graphs. The adequate relationship consists of (x,w) where x is a pair of graphs with n nodes and w is an isomorphism between them. Rerandomization generates a random permutation s and applies it to the pair of graphs or computes $s \circ w \circ s^{-1}$, respectively.

A.1. *Zero-Knowledge Proof of Knowledge*

Figure A.2 shows a 5-round zero-knowledge proof of knowledge of a solution of an arbitrary random self-reducible problem, a generalized version of the protocol by Bellare et al. [2]. $\overset{?}{=}$ and $\overset{?}{\in}$ denote that a party checks equality or set membership, respectively, and aborts if the check fails.

P starts by generating a new random instance x_P of the problem and sends it to V (1), which commits to his challenges using x_P (2). P then sends n rerandomizations of the to-be-proven instance (3), whereupon V opens the commitments to his challenges (4). P checks the openings, answers the challenges by either sending his rerandomization bits or the rerandomized witness, and also sends the witness of the instance generated in the first step (5). Finally, V checks the answer and either accepts or rejects.

Compared to the protocol for our framework, P only proves the knowledge of one witness w instead of a vector of witnesses $\mathbf{w}$. Similarly, the commitment uses only one basic instance (x_P, w_P), and the vectors sent correspond to n independent instances of the basic protocol in Fig. A.1.

A.1.1. *Zero-Knowledge*

The proof for protocols for random self-reducible problems is basically the same as in Sect. 5. Therefore, the test algorithm resembles the one presented earlier:

1. Take x_P as input and send it to V^*.
2. Wait for the commitments from V^*.
3. Generate $a'_1,\dots,a'_n$ as an honest prover would and send them to V^*.
4. Wait for $e'_1,\dots,e'_n$ and the opening information from V^*. Check the commitments. If the check fails, return `fail`, otherwise, continue.

P Input: $(x, w) \in R$		V Input: x	
$(x_P, w_P) \in_R R$			
	$\xrightarrow{\quad x_P \quad}$		(1)
		$e_1, \dots, e_n \in_R \{0, 1\}$ $\forall i : \begin{cases} u_i \in_R \{0,1\}^{O(n)} & e_i = 0 \\ (x_i, w_i) \in_R R & e_i = 1 \end{cases}$ $\forall i \ c_i := \begin{cases} \rho_x(x_P, u_i) & e_i = 0 \\ x_i & e_i = 1 \end{cases}$	
	$\xleftarrow{\quad c_1, \dots, c_n \quad}$		(2)
$r_i \in_R \{0, 1\}^{O(n)}$ $a_i := \rho_x(x, r_i)$			
	$\xrightarrow{\quad a_1, \dots, a_n \quad}$		(3)
		$\forall i \ d_i := \begin{cases} u_i & e_i = 0 \\ w_i & e_i = 1 \end{cases}$	
	$\xleftarrow{\quad e_1, \dots, e_n, d_1, \dots, d_n \quad}$		(4)
$\forall i : \begin{cases} c_i \overset{?}{=} \rho_x(x_P, d_i) & e_i = 0 \\ (c_i, d_i) \overset{?}{\in} R & e_i = 1 \end{cases}$ $\forall i \ z_i := \begin{cases} r_i & e_i = 0 \\ \rho_w(w, r_i) & e_i = 1 \end{cases}$			
	$\xrightarrow{\quad z_1, \dots, z_n, w_P \quad}$		(5)
		$(x_P, w_P) \overset{?}{\in} R$ $\forall i : \begin{cases} a_i \overset{?}{=} \rho_x(x, z_i) & e_i = 0 \\ (a_i, z_i) \overset{?}{\in} R & e_i = 1 \end{cases}$	

Fig. A.2. Zero-knowledge proof of knowledge of a solution of a random self-reducible problem.

5. Rewind V^* to the state before step 3, sample valid answers to the challenges $e'_1, \dots, e'_n$ ($(a_n, z_n) \in_R R$ or rerandomizations of x), compute the matching $a_1, \dots, a_n$ (rerandomization of x if necessary), and send them to V^*.
6. Wait for $e_1, \dots, e_n$ and the opening information from V^*. Check the commitments. If the check fails, repeat from step 5. Otherwise, return all messages from V^* and $a_1, \dots, a_n, z_1, \dots, z_n$.

Witness extraction requires the commitments $c_1, \dots, c_n$, two different challenge vectors $e_1, \dots, e_n$ and $e'_1, \dots, e'_n$ together with their opening information $d_1, \dots, d_n$ and $d'_1, \dots, d'_n$. Let i the index where the vectors differ: $e_i \neq e'_i$. Without loss of generality, we can assume that $e_i = 0$ and $e'_i = 1$. Therefore, $c_i = \rho_x(x_P, d_i)$ and $(c_i, d'_i) \in R$. We follow that $\rho_w^{-1}(d'_i, d_i)$ is a witness of x_P.

The simulator M_{V^*} then works as follows:

1. Fix the randomness of V^*.
2. Call the test algorithm with x_P as a rerandomized version of x, i.e., $x_P := \rho_x(x, r)$, $r \in_R \{0, 1\}^{O(n)}$.
    - If it returns `fail`, return `fail` as well.

- If it returns two different challenge vectors, use the witness extraction algorithm to compute w. Then, execute the following loop:
  (a) Generate $(x_P, w_P) \in_R R$.
  (b) Rewind V^* to the beginning.
  (c) Call the test algorithm with x_P. If it returns twice the same challenge vector, continue with the loop. Otherwise, output the conversation in the second run of the rerandomizer extraction, and x_P, the commitments and the second vector together with its opening information from the test algorithm. Moreover, compute $z_1, \dots, z_n$ using w, output them together with w_P, and stop.
- If it returns twice the same challenge, execute the following loop:
  (a) Generate $(x_P, w_P) \in_R R$.
  (b) Rewind V^* to the beginning.
  (c) Call the test algorithm with x_P. If it returns two different challenge vectors, continue with the loop. Otherwise, output the messages of the second round of the rerandomizer extraction algorithm, the output of the test algorithm, and $z_1, \dots, z_n$ generated by the test algorithm, and stop.

In the same way as in Sect. 5 and as Bellare et al. [2], one can argue that the simulator runs in polynomial time and that the generated transcript has the same distribution as a real execution of the protocol.

A.1.2. *Knowledge Extraction*

The knowledge extractor from Sect. 5 can be used almost identically here. The only difference is the generation of equivocable commitments: For $i = 1, \dots, n$, choose $u_i \in_R \{0,1\}^{O(n)}$ and compute $c_i := \rho_x(x_P, u_i)$. To open, let

$$d_i := \begin{cases} u_i, & e_i = 0, \\ \rho_w(w_P, u_i), & e_i = 1. \end{cases}$$

It can easily be seen that such a commitment and the opening are distributed in the same way as a regular commitment. For $e_i = 0$, the commitment is generated in the exact same manner, and for $e_i = 1$, $(\rho_x(w_P, u_i), \rho_w(w_P, u_i))$ is distributed uniformly by the definition of random self-reducible problems.

In this vein, one can prove that the protocol in Fig. A.2 is a proof of knowledge with knowledge error 2^{1-n} because the challenge space $\{0,1\}^n$ has size 2^n instead of $|\Omega_n|$.

References

[1] M. Bellare, O. Goldreich, On probabilistic versus deterministic provers in the definition of proofs of knowledge. *Electron. Colloq. Comput. Complex.* **13**(136) (2006)

[2] M. Bellare, S. Micali, R. Ostrovsky, Perfect zero-knowledge in constant rounds, in *STOC* (ACM, New York, 1990), pp. 482–493

[3] R. Bendlin, I. Damgård, C. Orlandi, S. Zakarias, Semi-homomorphic encryption and multiparty computation, in *EUROCRYPT*, ed. by K.G. Paterson. Lecture Notes in Computer Science, vol. 6632 (Springer, Berlin, 2011), pp. 169–188

[4] M. Blum, Coin flipping by telephone, in *CRYPTO* (1981), pp. 11–15

[5] I. Damgård, M. Geisler, M. Krøigaard, Efficient and secure comparison for on-line auctions, in *ACISP*, ed. by J. Pieprzyk, H. Ghodosi, E. Dawson. Lecture Notes in Computer Science, vol. 4586 (Springer, Berlin, 2007), pp. 416–430

[6] A. Fiat, A. Shamir, How to prove yourself: practical solutions to identification and signature problems, in *CRYPTO*, ed. by A.M. Odlyzko. Lecture Notes in Computer Science, vol. 263 (Springer, Berlin, 1986), pp. 186–194

[7] E. Fujisaki, T. Okamoto, Statistical zero knowledge protocols to prove modular polynomial relations, in *CRYPTO*, ed. by B.S. Kaliski Jr. Lecture Notes in Computer Science, vol. 1294 (Springer, Berlin, 1997), pp. 16–30

[8] S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof-systems (extended abstract), in *STOC* (ACM, New York, 1985), pp. 291–304

[9] J. Groth, Cryptography in subgroups of z_n, in *TCC*, ed. by J. Kilian. Lecture Notes in Computer Science, vol. 3378 (Springer, Berlin, 2005), pp. 50–65

[10] Y. Ishai, E. Kushilevitz, R. Ostrovsky, A. Sahai, Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.* **39**(3), 1121–1152 (2009)

[11] C.-P. Schnorr, Efficient signature generation by smart cards. *J. Cryptol.* **4**(3), 161–174 (1991)