Des. Codes Cryptogr. (2015) 77:663–676
DOI 10.1007/s10623-015-0103-5

Zero-knowledge proofs of knowledge for group homomorphisms

Ueli Maurer[1]

Received: 13 October 2014 / Revised: 23 May 2015 / Accepted: 26 May 2015 /
Published online: 3 June 2015

Abstract A simple zero-knowledge proof of knowledge protocol is presented of which many known protocols are instantiations. These include Schnorr's protocol for proving knowledge of a discrete logarithm, the Fiat–Shamir and Guillou–Quisquater protocols for proving knowledge of a modular root, protocols for proving knowledge of representations (like Okamoto's protocol), protocols for proving equality of secret values, a protocol for proving the correctness of a Diffie–Hellman key, protocols for proving the multiplicative relation of three commitments (as required in secure multi-party computation), and protocols used in credential systems. This unifies a substantial body of work and can also lead to instantiations of the protocol for new applications.

Keywords Zero-knowledge protocol · Proof of knowledge · Group homomorphism

Mathematics Subject Classification 94A60

1 Introduction

Finite cyclic groups, in particular elliptic curves over a finite field, are of paramount importance in cryptography. A large number of known cryptographic systems and protocols is based on the hardness of a certain problem defined for such a group, most prominently the discrete logarithm problem. Scott Vanstone was one of the prime contributors to this field, and this paper, which presents a zero-knowledge proof of knowledge of a pre-image in a

This is one of several papers published in *Designs, Codes and Cryptography* comprising the "Special Issue on Cryptography, Codes, Designs and Finite Fields: In Memory of Scott A. Vanstone".

A preliminary version of this paper appeared at AFRICACRYPT 2009 [14].

✉ Ueli Maurer
maurer@inf.ethz.ch

[1] Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland

finite group, is dedicated to Scott, with whom I have also had the pleasure of collaborating fruitfully [13] and who has been a dear friend for almost three decades.

1.1 Interactive proofs

A conventional proof of a statement is, informally, a sequence of elementary, easily verifiable steps, which, starting from the axioms (or previously proven facts), in the last step yields the statement to be proved. In contrast to a conventional proof, an *interactive proof* [11] is a protocol that is defined between a *prover*, usually called P or Peggy, and a *verifier*, usually called V or Vic. More formally, an interactive proof is a pair (P, V) of programs implementing the protocol steps Peggy and Vic are supposed to execute. An interactive proof must be complete and sound. Completeness means that an honest prover succeeds in convincing an honest verifier, and soundness means that a dishonest prover does not succeed in convincing Vic of a false statement.

Several motivations, theoretical and practical, exist for considering interactive proofs as opposed to conventional proofs. One main motivation is that, in contrast to a conventional proof, an interactive proof can be performed in a way that transfers only the conviction that the claimed statement is true but does not leak any further information, in particular not a transferable proof. More precisely, an interactive proof is called *zero-knowledge* if the verifier could simulate the entire protocol transcript by himself, without interacting with the prover. In particular, this implies that the transcript is not convincing for any other party.

There are two types of interactive proofs: proofs of a mathematical statement and proofs of knowledge. A proof of knowledge proves that Peggy knows a value satisfying a certain predicate (a witness). Often a proof of a mathematical statement (e.g. that a number is a square modulo an RSA modulus) is carried out as a proof of knowledge of a witness for the statement (i.e., of a square root).

1.2 Contributions of this paper

We introduce a new level of abstraction in a general type of proof of knowledge, namely of a preimage of a group homomorphism, and thereby unify and generalize a large number of protocols in the literature. We observe that actually the identical principle has been reused several times, where each result came with a separate proof. While the similarity of different protocols certainly did not go unnoticed, a viewpoint that shows them to be instantiations of the *same* protocol is new to our knowledge. We call this protocol, described in Sect. 5, the main protocol.

The relation of our results to Cramer's Σ-protocols [6] (see also [7]), also an abstraction of a general type of proofs of knowledge, will be discussed in Sect. 5.3. In short, we go a step further and not only abstract a protocol *type*, but actually show that many protocols are the *same* protocol when seen at the right level of abstraction.

The advantage of our abstract viewpoint is that one can provide a proof once and for all, and for each individual instantiation only needs to describe the group homomorphism underlying the particular example and check the conditions of Theorem 3, our main theorem. This requires just a few lines for a complete proof that a given protocol is a zero-knowledge proof of knowledge. Moreover, this approach leads to new protocols by using new instantiations of the group homomorphism.

1.3 Some terminology

We use the standard notions of efficient and negligible and point out that such definitions are asymptotic, i.e., for asymptotic families (of groups) depending on a security parameter (which we will not make explicit). Efficient is usually defined as polynomial-time and negligible as vanishing faster than the inverse of any polynomial. In general, we assume that the reader is familiar with the basic aspects of interactive proofs and cryptographic thinking.

1.4 Outline

The outline of the paper is as follows. In Sect. 2 we discuss two well-known examples of interactive proofs. In Sect. 3 we formalize the concept of a proof of knowledge. In Sect. 4 we define what it means for a protocol to be zero-knowledge. In Sect. 5 we present our new general protocol, referred to as the *main protocol*, and prove that it is a zero-knowledge proof of knowledge. In Sect. 6 we show that many known and new protocols are instantiations of the main protocol. Sects. 3 and 4 can be skipped if the reader is familiar with the topic and is just interested in the unified viewpoint.

2 Two protocol examples

In this section we discuss two classical examples of interactive proofs of knowledge.

2.1 The Schnorr protocol

Consider a cyclic group H with prime order $|H| = q$ for which computing discrete logarithms (DL) is considered infeasible. Peggy wants to prove to Vic that she knows the DL x of an element z to the base h, i.e., that she knows x such that $z = h^x$. For example, z could be Peggy's public key and the protocol is then used as an identification protocol by which Peggy proves knowledge of the secret key.

The protocol, proposed by Schnorr [18], works as follows (see Fig. 1). First, Peggy chooses $k \in \mathbf{Z}_q$ at random and sends the group element $t = h^k$ to Peggy. Then Vic chooses a challenge value $c \in \mathcal{C}$ at random from a challenge space $\mathcal{C}$ which is a subset of $[0, q-1]$. Then Peggy answers by sending the value $r = k + xc \pmod q$. Finally, Vic accepts the protocol execution if and only if $h^r = t \cdot z^c$.

Let us analyze the protocol. It is easy to see that if Peggy knows x and performs the protocol honestly, then Vic will accept (completeness). To argue about soundness, we observe that unless Peggy knows x, she cannot answer more than one challenge correctly. This can be seen as follows. If Peggy could answer, for a fixed t, two challenges c and c' by r and r', respectively, so that Vic accepts, then she could compute x (and hence knows it).[1] This can be shown as follows: we have $h^r = t \cdot z^c$ and $h^{r'} = t \cdot z^{c'}$ and thus

$$h^{r-r'} = z^{c-c'} = h^{x(c-c')}.$$

Therefore,

$$r - r' \equiv x(c - c') \pmod q,$$

[1] A correct argument is more involved; one has to argue that there exists an efficient knowledge extractor (see Sect. 3).

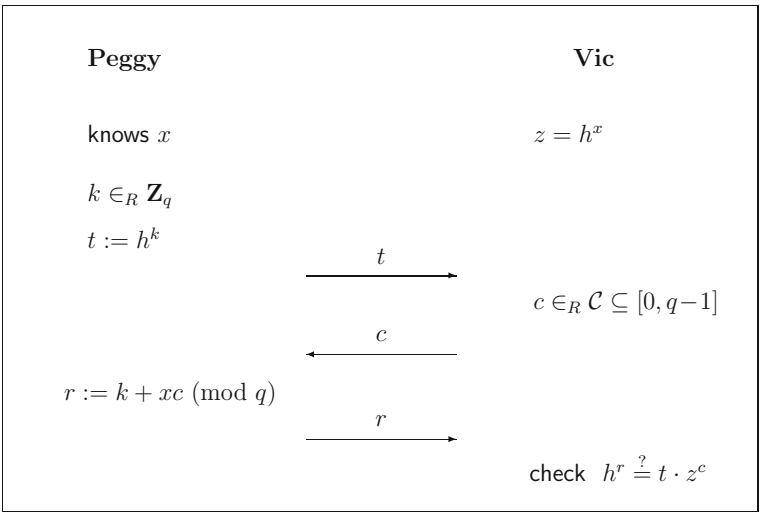

Fig. 1 The Schnorr protocol for proving knowledge of a discrete logarithm x of an element $z = h^x$ in the group H

from which we obtain

$$x \equiv \frac{r_1 - r_2}{c_1 - c_2} \pmod q.$$

Note that it is important that q is prime since otherwise the inverse of $c_1 - c_2$ modulo q may not be defined, unless one restricts $\mathcal{C}$ in an artificial way.

One might be tempted to conclude from the above argument that any prover with success probability at least $2/|\mathcal{C}|$ can answer at least two challenges and therefore knows x. However, this informal argument turns out to be incorrect (see Sect. 3), requiring a clean formalization of what it means to *know* a value.

Now we argue informally that the protocol is zero-knowledge. Without knowledge of x, one can, for any challenge c, generate a triple (t, c, r) with the distribution as it occurs in the protocol. One can prove (this is not entirely trivial) that even a dishonest verifier can simulate perfectly the entire transcript he would see in a protocol execution with Peggy, i.e., with the same probability distribution as it occurs in the real protocol (see Sect. 4). However, the simulation is efficient only if the size $|\mathcal{C}|$ of the challenge space is bounded to polynomial size. To obtain the zero-knowledge property, one may therefore choose $|\mathcal{C}|$ to be relatively small (e.g. on the order of 10^6), and repeat the protocol several (say s) times. Such a protocol is zero-knowledge and achieves the soundness guarantees corresponding to the size of the overall challenge space $\mathcal{C}^s$.

2.2 The Fiat–Shamir and Guillou–Quisquater protocols

Consider a modulus m which is assumed to be difficult to factor. For concreteness, one can think of m as being an RSA-modulus [17]. For a given exponent e (with $\gcd(e, \varphi(m)) = 1$), breaking the RSA cryptosystem means to compute eth roots modulo m. This is considered hard and, for a generic model of computation, has been proved to be equivalent to factoring m [1]. Unlike for RSA, in our context e is considered to be prime.

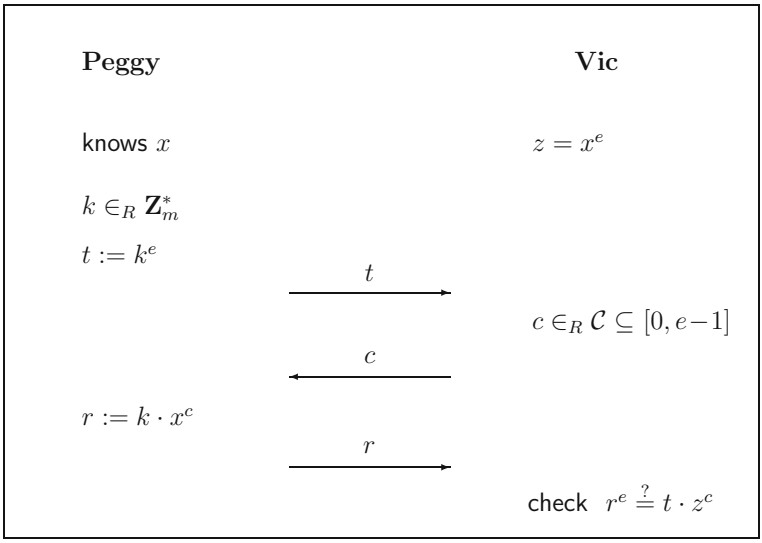

Fig. 2 The Guillou–Quisquater (GQ) protocol for proving knowledge of an eth root x modulo m of a given element $z \in \mathbf{Z}_m^*$

The Guillou–Quisquater (GQ) protocol [12] allows Peggy to prove to Vic that she knows the eth root x modulo m of a given number $z \in \mathbf{Z}_m^*$, i.e., she knows x such that $x^e = z$ in $\mathbf{Z}_m^*$. (Again, z could be Peggy's public key for which she wants to prove knowledge of the corresponding private key.)

The protocol works as follows (see Fig. 2). First, Peggy chooses $k \in \mathbf{Z}_m^*$ at random and sends the group element $t = k^e$ to Peggy. Then Vic chooses a challenge value $c \in \mathcal{C}$ at random from a challenge space $\mathcal{C} \subseteq [0, e-1]$. Then Peggy answers by sending the value $r = k \cdot x^c$ (in $\mathbf{Z}_m^*$, i.e., modulo m). Finally, Vic accepts the protocol execution if and only if $r^e = t \cdot z^c$.

This protocol is a generalization of the Fiat–Shamir protocol [9,10] which considers the special case $e = 2$. If $|\mathcal{C}|$ (and hence e) is sufficiently large, then a single execution of the protocol suffices. Otherwise, the protocol is repeated a sufficient number of times.

It is easy to see that if Peggy knows x and performs the protocol honestly, then Vic will accept (correctness). To argue about soundness, we observe again that unless Peggy knows x, she cannot answer more than one challenge correctly. This can be seen as follows. If Peggy could answer, for fixed t, both challenges c and c' by r and r', respectively (so that Vic accepts), then she could compute x (and hence knows it). This can be shown as follows: We have (in $\mathbf{Z}_m^*$) $r^e = t \cdot z^c$ and $r'^e = t \cdot z^{c'}$ and thus

$$\left(\frac{r}{r'}\right)^e \equiv z^{c-c'} \pmod{m}.$$

and hence

$$\frac{r}{r'} \equiv x^{c-c'} \pmod{m}.$$

In addition to $x^{c-c'}$, Peggy trivially knows another power of x, namely $z = x^e$. When e is prime, then $c - c'$ and e are relatively prime. From two different powers of x with relatively

prime exponents one can compute x. Namely, application of Euclid's extended gcd-algorithm yields integers a and b such that

$$ea + (c - c')b = 1.$$

Therefore x can be computed as $x = x^{ea+(c-c')b}$, i.e., as

$$x \equiv z^a \cdot \left(\frac{r}{r'}\right)^b \pmod{m}.$$

2.3 Comparing the two protocols

The Schnorr protocol and the GQ protocol have a number of similarities, as already indicated by the fact that we chose to use the same names (k, t, c, and r) for the different quantities appearing in the protocol. But nevertheless the two protocols are quite different. The mathematical structure is different and so is the argument for proving that from the answers to two different challenges one can compute x. However, in Sects. 5 and 6 we show that there is a level of abstraction at which the two protocols are identical, i.e., instantiations of the same protocol, as are many more protocols proposed in the literature.

3 Proofs of knowledge

In this section we recall the definition of a proof of knowledge due to Feige et al. [9] and state a general theorem that can be used to easily prove that a protocol is a proof of knowledge.

The above soundness argument, namely that being able to answer two challenges implies knowledge of the secret value x, must be made more precise. Let us formalize the concept of a proof of knowledge. What constitutes knowledge, corresponding to a given value z, is defined by a (verification) predicate[2]

$$Q : \{0,1\}^* \times \{0,1\}^* \to \{\texttt{false}, \texttt{true}\}.$$

For a given value (a bit-string) z, Peggy claims to know a value (bit-string) x such that $Q(z, x) = \texttt{true}$.

The following classical definition' [9] captures the notion that being successful in the protocol implies knowledge of a witness x with $Q(z, x) = \texttt{true}$.

Definition 1 An interactive protocol (P, V) is a *proof of knowledge* for predicate Q if the following holds:

- (Completeness.) V accepts when P has as input an x with $Q(z, x) = \texttt{true}$.
- (Soundness.) There is an efficient program K, called *knowledge extractor*, with the following property. For any (possibly dishonest) $\hat{P}$ with non-negligible probability of making V accept, K can interact with $\hat{P}$ and outputs (with overwhelming probability) an x such that $Q(z, x) = \texttt{true}$.[3]

We now capture the special property of a protocol which we proved for the Schnorr and the GQ protocols and which allowed us to argue about soundness.

[2] Equivalently, one can consider a relation on $\{0,1\}^*$.

[3] K must be able to choose the randomness of $\hat{P}$ and to reset $\hat{P}$.

Definition 2 Consider a predicate Q for a proof of knowledge. A three-move protocol round (Peggy sends t, Vic sends c, Peggy sends r) with challenge space $\mathcal{C}$ is 2-*extractable*[4] if from any two triples (t, c, r) and (t, c', r') with distinct $c, c' \in \mathcal{C}$ accepted by Vic one can efficiently compute an x with $Q(z, x) = \texttt{true}$.

The following theorem (see also [6,7]) states that for a protocol to be a proof of knowledge it suffices to check the 2-extractability condition for one (three-move) round of the protocol.

Theorem 1 *An interactive protocol consisting of s 2-extractable rounds with challenge space $\mathcal{C}$ is a proof of knowledge for predicate Q if $1/|\mathcal{C}|^s$ is negligible.*[5]

Proof We need to exhibit a knowledge extractor K. It can be defined by the following simple procedure:

1. Choose the randomness for $\hat{P}$.
2. Generate two independent protocol executions between $\hat{P}$ and V (with the *same* chosen randomness for $\hat{P}$).
3. If V accepts in both executions and the challenge sequences were distinct, then identify the first round with different challenges c and c' (but, of course, the same t). Use 2-extractability to compute an x, and output it (and stop).
   Otherwise go back to step 1.

It is not very difficult to show that the expected running time of the knowledge extractor is polynomial if the success probability of $\hat{P}$ is non-negligible. □

4 Zero-knowledge protocols

We now discuss the zero-knowledge property of a protocol. Informally, a protocol between P and V is zero-knowledge if even a dishonest V, which for this reason we call $\hat{V}$, does not learn anything from the protocol execution which he did not know before. This is captured by the notion of simulation [11]: $\hat{V}$ could simulate a protocol transcript by himself which is indistinguishable from a real transcript that would occur in an actual protocol execution between P and $\hat{V}$.

Definition 3 A protocol (P, V) is *zero-knowledge* if for every efficient program $\hat{V}$ there exists an efficient program S, the *simulator*, such that the output of S is indistinguishable from the view of $\hat{V}$ (consisting of its internal randomness and the transcript of the protocol execution between P and $\hat{V}$). If the indistinguishability is perfect,[6] i.e., the probability distribution of the simulated and the actual transcript are identical, then the protocol is called *perfect zero-knowledge*.

We now capture the special property of a protocol round, called c-simulatability, which is required to construct the zero-knowledge simulator.

Definition 4 A three-move protocol round (Peggy sends t, Vic sends c, Peggy sends r) with challenge space $\mathcal{C}$ is *c-simulatable*[7] if for any value $c \in \mathcal{C}$ one can efficiently generate a triple

[4] It is also often called special soundness [6,7] when the challenge space is large.

[5] The last point implies that every particular challenge sequence $c_1, \ldots, c_s$ has negligible probability of being selected by an honest verifier.

[6] The indistinguishability could also be statistical or computational.

[7] This is sometimes also called *special honest-verifier zero-knowledge* [6,7].

(t, c, r) with the same distribution as occurring in the protocol (conditioned on the challenge being c).

The following theorem states that for a protocol to be zero-knowledge it suffices to check the c-simulatability condition for one round of the protocol.

Theorem 2 *A protocol consisting of c-simulatable three-move rounds, with uniformly chosen challenge from a polynomially-bounded (per-round) challenge space $\mathcal{C}$, is perfect zero-knowledge.*

The proof of this theorem is not entirely trivial. We just describe the basic idea of the simulator. It simulates one round after the next. In each round (say the ith), the simulator chooses a uniformly random challenge c_i, generates a triple (t_i, c_i, r_i) using the c-simulatability, and then checks whether $\hat{V}$ would actually issue challenge c_i if it were in the corresponding state in round i. If the check succeeds, then this round is appended to the simulated transcript as the ith round, otherwise the simulation of the ith round is restarted.[8]

5 Proving knowledge of a preimage of a group homomorphism

5.1 One-way group homomorphisms

We consider two groups $(G, \star)$ and $(H, \otimes)$, where we intentionally use special symbols for the group operations, avoiding the addition and multiplication symbols "+" and "·". We assume that the group operations $\star$ and $\otimes$ are efficiently computable.

A function $f : G \to H$ is a homomorphism if

$$f(x \star y) = f(x) \otimes f(y).$$

We will consider the case where f is (believed to be) a one-way function, such that it is infeasible to compute x from $f(x)$ for a randomly chosen x.[9] In this case it is meaningful for a prover Peggy to prove that she knows an x such that for a given value z we have $z = f(x)$. To simplify the notation we write $[x]$ instead of $f(x)$.[10] We can consider $[x]$ to be an embedding of $x \in G$ in H. We point out that f need not be bijective and therefore a value $z = [x]$ does not necessarily determine x. But it is well-defined for which values x we have $z = [x]$.

Given embedded values $[x]$ and $[y]$ we can efficiently compute

$$[x \star y] = [x] \otimes [y],$$

without knowing x or y, due to the homomorphism.

5.2 The main protocol

The protocol in Fig. 3, which we call the main protocol, is a proof of knowledge of a value x such that $z = [x]$, for a given value z, provided that the two conditions stated in the following

[8] This requires access to the strategy of $\hat{V}$, or $\hat{V}$ must be rewindable.

[9] Note, however, that our treatment and claims do not depend on the one-way property. Should f not be one-way, then the protocols are perhaps less useful, but they still have the claimed properties.

[10] When we define a group homomorphism in terms of a given group homomorphism (denoted $[\cdot]$), then we write $[\![\cdot]\!]$ to avoid overloading the symbol $[\cdot]$.

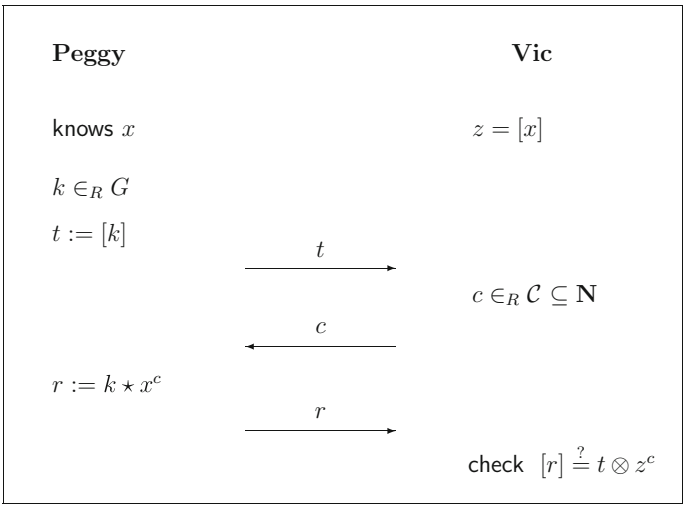

Fig. 3 Main protocol: proof of knowledge, for a given value z, of a value x such that $z = [x]$, where $x \mapsto [x]$ is a (one-way) group homomorphism

theorem are satisfied. Note that G and H need not be commutative. The challenge space can be chosen as an arbitrary subset of $\mathbf{N}$. If it is chosen to be small,[11] then one needs several (three-move) rounds to reduce the soundness error to be negligible.

Theorem 3 *If values $\ell \in \mathbf{Z}$ and $u \in G$ are known such that*

(1) $\gcd(c_1 - c_2, \ell) = 1$ *for all $c_1, c_2 \in \mathcal{C}$ (with $c_1 \neq c_2$), and*
(2) $[u] = z^\ell$,

then the three-move protocol round described in Fig. 3 is 2-extractable. Moreover, a protocol consisting of s rounds is a proof of knowledge if $1/|\mathcal{C}|^s$ is negligible, and it is zero-knowledge if $|\mathcal{C}|$ is polynomially bounded.

Proof 2-extractability can be proved as follows: From r and r' such that $[r] = t \otimes z^c$ and $[r'] = t \otimes z^{c'}$ for two different challenges c and c' we can obtain $\tilde{x}$ satisfying $[\tilde{x}] = z$, as

$$\tilde{x} = u^a \star \left(r'^{-1} \star r\right)^b,$$

where a and b are computed using Euclid's extended gcd-algorithm such that

$$\ell a + (c - c')b = 1.$$

We make use of

$$[r'^{-1} \star r] = [r'^{-1}] \otimes [r] = z^{-c'} \otimes t^{-1} \otimes t \otimes z^c = z^{-c'} \otimes z^c = z^{c-c'}$$

[11] There can be at least two reasons for choosing a small challenge space. First, a larger space may not work, for example if e is small in the GQ protocol. Second, one may want the protocol to be zero-knowledge, which generally does not hold for large (per-round) challenge space.

to see that $[\tilde{x}] = z$:

$$\begin{aligned}[\tilde{x}] &= [u^a \star (r'^{-1} \star r)^b] \\ &= [u]^a \otimes [r'^{-1} \star r]^b \\ &= (z^\ell)^a \otimes (z^{c-c'})^b \\ &= z^{\ell a+(c-c')b} \\ &= z.\end{aligned}$$

Theorem 1 implies directly that the protocol is a proof of knowledge, and Theorem 2 implies that it is zero-knowledge if $|\mathcal{C}|$ is polynomially bounded since it is c-simulatable. This is easy to see: Given z and a challenge c, one chooses r at random and computes t as $t = [r] \otimes z^{-c}$. □

5.3 Comparison with some related work

The approach described in this paper is similar in flavor to other papers, some of which also consider a more general case where the group order is unknown (e.g. see [2–5]) but none of them achieves the presented level of abstraction. The result should be contrasted with another approach, due to Cramer [6] (see also [7]) to abstracting a general type of proofs of knowledge. Cramer introduced the notion of Σ-protocols which are basically three-move protocols, as discussed in this paper, which are both 2-extractable and c-simulatable. All the protocols we consider are Σ-protocols. However, we go further in that we show that a large class of protocols are not only of the same protocol type, but are actually the *same protocol*, thus requiring only one proof of the claimed properties. In order to apply our Theorem 3 one only needs to specify the groups G and H, the homomorphism, and check the two conditions of Theorem 3.

6 Special cases of the main protocol

In this section we describe a number of protocols as instantiations of our main protocol.

6.1 Schnorr and GQ as special cases

The Schnorr protocol is the special case where $(G, \star) = (\mathbf{Z}_q, +)$ (with addition modulo q in $\mathbf{Z}_q$) and H is a group of order q with the group operation written as multiplication (i.e., "$\cdot$", which can also be omitted). The (one-way) group homomorphism is defined by

$$G \to H : \quad x \mapsto [x] = h^x.$$

The challenge space $\mathcal{C}$ can be an arbitrary subset of $[0, q-1]$. The two conditions of Theorem 3 are satisfied for $\ell = q$ (if q is prime) and $u = 0$. Note that $\gcd(c_1 - c_2, \ell) = 1$ for all distinct $c_1, c_2 \in \mathcal{C}$, and $[u] = [0] = 1 = z^\ell$ since every element of H raised to the group order $|H| = q$ is the neutral element of H.

The GQ protocol is the special case where $(G, \star) = (\mathbf{Z}_m^*, \cdot) = (H, \otimes)$. The one-way homomorphism is defined by

$$G \to H : \quad x \mapsto [x] = x^e.$$

The challenge space $\mathcal{C}$ can be an arbitrary subset of $[0, e-1]$, provided e is prime. The conditions of Theorem 3 are satisfied for $\ell = e$ and $u = z$. Note that $\gcd(c_1 - c_2, \ell) = 1$ for all distinct $c_1, c_2 \in \mathcal{C}$, and $[u] = [z] = z^e = z^\ell$.

6.2 Proof of knowledge of several values

Let

$$G_i \to H_i : \ x \mapsto [x]^{(i)}$$

for $i = 1, \ldots, n$ be (possibly distinct) group homomorphisms for which, for the same ℓ, there exist $u_1, \ldots, u_n$ and $z_1, \ldots, z_n$ satisfying Condition (2) in Theorem 3, i.e., $[u_i]^{(i)} = z_i^\ell$ for $i = 1, \ldots, n$. Then also

$$\begin{aligned} &G_1 \times \cdots \times G_n \to H_1 \times \cdots \times H_n : \\ &(x_1, \ldots, x_n) \mapsto [(x_1, \ldots, x_n)] = \left([x_1]^{(1)}, \ldots, [x_n]^{(n)}\right) \end{aligned}$$

is a one-way group homomorphisms.[12] Therefore the main protocol proves in one stroke the knowledge of $x_1, \ldots, x_n$ such that for given $z_1 \in H_1, \ldots, z_k \in H_n$ we have $z_1 = [x_1]^{(1)}, \ldots, z_n = [x_n]^{(n)}$. This can be seen by setting $u = (u_1, \ldots, u_n)$ and $z = (z_1, \ldots, z_n)$ since

$$[u] = \left([u_1]^{(1)}, \ldots, [u_n]^{(n)}\right) = \left(z_1^\ell, \ldots, z_n^\ell\right) = z^\ell.$$

A typical application of this protocol is for proving knowledge of several discrete logarithms in (possibly distinct) groups of prime order q.

6.3 Proof of equality of embedded values

Let again

$$G \to H_i : \ x \mapsto [x]^{(i)}$$

for $i = 1, \ldots, n$ be one-way group homomorphisms as in the previous section, but with $u_1 = \cdots = u_n = u$. Then also

$$G \to H_1 \times \cdots \times H_n : \ x \mapsto [x] = \left([x]^{(1)}, \ldots, [x]^{(n)}\right)$$

is a group homomorphisms (but not necessarily one-way). Therefore the main protocol proves the knowledge of x that is simultaneously a preimage of all n homomorphisms. More precisely, it proves knowledge of x such that for given $z_1 \in H_1, \ldots, z_k \in H_k$ we have $z_1 = [x]^{(1)}, \ldots, z_k = [x]^{(k)}$.[13] This can be seen by setting $z = (z_1, \ldots, z_n)$. A typical application of this protocol is for proving that several discrete logarithms in groups of prime order q are identical.

6.4 Proof of knowledge of a representation

Consider again a group H with prime order q, and let several generators $h_1, \ldots, h_m$ of H be given. A representation of an element $z \in H$ is a list $(x_1, \ldots, x_m)$ of exponents such

[12] The group operations in $G_1 \times \cdots \times G_n$ and $H_1 \times \cdots \times H_n$ are defined component-wise.

[13] Note that if the homomorphisms are bijective, then this protocol not only proves knowledge of x, but actually that all embedded values are identical.

that $z = h_1^{x_1} h_2^{x_2} \cdots h_m^{x_m}$. (Note that such a representation is not unique.) We want to prove knowledge of a representation of a given element z.

For the special case $m = 2$, a protocol for this purpose was proposed by Okamoto [15]. This is of interest, among other reasons, since Pedersen commitments [16] have this form.[14]

A protocol for proving knowledge of a representation can be obtained as another simple instantiation of our main protocol, using the homomorphism

$$\mathbf{Z}_q^m \to H: \ (x_1, \ldots, x_m) \mapsto [(x_1, \ldots, x_m)] = h_1^{x_1} \cdots h_m^{x_m}.$$

The conditions of Theorem 3 are satisfied for the choice $\ell = q$ and $u := (0, \ldots, 0)$ since $[(0, \ldots, 0)] = h_1^0 \cdots h_m^0 = 1 = z^\ell$ for every $z \in H$.

6.5 Proof of knowledge of a set of linear representations

One can actually prove more general statements about the knowledge of representations, namely knowledge of values $x_1, \ldots, x_r$ that simultaneously satisfy several representation equations with respect to generators $h_1, \ldots, h_m$. Such protocols appear, for example, in the literature on credential systems.

For example, consider generators h_1, h_2, h_3 of H. For given values $z_1, z_2 \in H$ we can prove knowledge of values $x_1, x_2, x_3, x_4 \in \mathbf{Z}_q$ satisfying $z_1 = h_1^{x_3} h_2^{x_1}$ and $z_2 = h_1^{x_2} h_2^{x_4} h_3^{x_1}$. The reader can figure out as an exercise how the homomorphism must be chosen such that our main protocol provides such a proof.

More generally, one can prove knowledge of $x_1, \ldots, x_r$ such that for given values $z_1, \ldots, z_s$ and, for sm linear (over $GF(q)$) functions $\phi_{11}, \ldots, \phi_{sm}$ from $GF(q)^r$ to $GF(q)$, we have

$$z_i = h_1^{\phi_{i1}(x_1, \ldots, x_r)} \cdot h_2^{\phi_{i2}(x_1, \ldots, x_r)} \cdots h_m^{\phi_{im}(x_1, \ldots, x_r)}$$

for $i = 1, \ldots, s$. The group homomorphism $\mathbf{Z}_q^r \to H^s$ is defined as

$$[(x_1, \ldots, x_r)] = \left(\prod_{j=1}^{m} h_j^{\phi_{1j}(x_1, \ldots, x_r)}, \ldots, \prod_{j=1}^{m} h_j^{\phi_{sj}(x_1, \ldots, x_r)} \right).$$

6.6 Proof of correctness of Diffie–Hellman keys

Let H be a group with prime order $|H| = q$ and generator h used in the Diffie–Hellman protocol [8]. As for the Schnorr protocol, we define a homomorphic embedding by

$$G \to H: \ x \mapsto [x] = h^x.$$

Recall that in the Diffie–Hellman protocol, Alice chooses an $a \in \mathbf{Z}_q$ and sends $[a] = h^a$ to Bob and, symmetrically, Bob chooses a $b \in \mathbf{Z}_q$ and sends $[b] = h^b$ to Alice. The common secret key is $[ab] = h^{ab}$. It is believed that for general groups it is computationally hard to decide whether or not a given key $K \in H$ is the correct key, i.e., whether $K = h^{ab}$. This is known as the Decisional Diffie–Hellman (DDH) problem. For example, if a very powerful organization were willing to compute Diffie–Hellman keys as a commercial service (returning $[ab]$ when given $[a]$ and $[b]$), then the customer could not verify that the key is correct. In this context, as well as in other contexts, it is useful to be able to prove the correctness of a

[14] One commits to a value x by choosing a random r and sending $h_1^x h_2^r$ as the commitment (see also Sect. 6.7). This commitment scheme is information-theoretically hiding (but only computationally binding).

Diffie–Hellman key in zero-knowledge, in particular without leaking any information about a or b. This is again achieved by a simple instantiation of our main protocol.

Let values $A = [a]$, $B = [b]$ and $C = [c]$ be given. We wish to prove that $c = ab \pmod q$, i.e., that A, B, and C form a so-called Diffie–Hellman triple. For this purpose we define the following one-way group homomorphism which we denote by $[\![\cdot]\!]$ and which is defined in terms of the homomorphism $[\cdot]$ and of B:

$$\mathbf{Z}_q \to H \times H : \ x \mapsto [\![x]\!] = ([x], [xb]) = (h^x, B^x).$$

Note that $[xb]$ can be computed efficiently from $B = [b]$ and x without knowing b. This yields, as a special case of the main protocol for the homomorphism $x \to [\![x]\!]$, the desired proof: One proves knowledge of a preimage x (namely $x = a$) such that

$$[\![x]\!] = (A, C).$$

Due to the particular choice of the homomorphism, this implies that $c = ab$.

While the protocol proves that the prover Peggy knows a, it does not prove that she knows b or c. (This does not contradict the fact that the claim $c = ab$ is indeed proved.) If desired, a proof of knowledge of b (and hence also of c) could be linked into the above proof using the technique of Sect. 6.3.

6.7 Multiplication proof for pedersen commitments

An important step in secure multi-party computation (MPC) protocols is for a party to commit to the product of two values it is already committed to, and to prove that this product commitment is correct. We show how such a proof can be given for Pedersen commitments.

Recall that in the Pedersen commitment scheme one commits to a value $x \in \mathbf{Z}_q$ by choosing $\rho_x \in \mathbf{Z}_q$ at random and sending the value $g^x h^{\rho_x}$. (To avoid unnecessary indices we denote here the two generators as g and h instead of h_1 and h_2.) We consider the commitment one-way homomorphism

$$\mathbf{Z}_q \times \mathbf{Z}_q \to H : \ (x, \rho_x) \mapsto [(x, \rho_x)] = g^x h^{\rho_x}.$$

Since this commitment scheme is information-theoretically hiding, the value x is not determined by the commitment. What counts is *how* the committing party can open the commitment.

Let three commitments $A, B, C \in H$ by Peggy be given. In the following we assume that it is clear from the context that Peggy can open B as (b, ρ_b). If this were not the case, one could incorporate such a proof using the technique of Sect. 6.3. We describe a protocol that allows Peggy to prove that she can open A as (a, ρ_a) and C as (c, ρ_c) with $c = ab$.

For this purpose we define the following one-way group homomorphism:

$$\mathbf{Z}_q^3 \to H \times H : \ (x, \rho_x, \sigma_x) \mapsto [\![(x, \rho_x, \sigma_x)]\!] = ([(x, \rho_x)], [(xb, x\rho_b + \sigma_x)]),$$

where the second component can be computed as

$$[(xb, x\rho_b + \sigma_x)] = B^x h^{\sigma_x}$$

without knowledge of b and ρ_b (with $B = [(b, \rho_b)]$).

The desired proof can now be obtained as a special case of the main protocol: Peggy proves that she knows a triple (x, ρ_x, σ_x) such that

$$[\![(x, \rho_x, \sigma_x)]\!] = (A, C).$$

As can easily be verified, this proof is successful for the choice

$$(x, \rho_x, \sigma_x) = (a, \rho_a, \rho_c - a\rho_b).$$

7 Conclusions

The described protocol for proving knowledge of a preimage of a group homomorphism is the abstraction of a large class of protocols. The presented list of examples is by no means exhaustive. We encourage the reader to find other protocols in the literature which can be described as an instance of the main protocol.

References

1. Aggarwal D., Maurer U.: Breaking RSA generically is equivalent to factoring. In: Advances in Cryptology—EUROCRYPT 2009. Lecture Notes in Computer Science, vol. 5479, pp. 36–53. Springer, Berlin (2009).
2. Bangerter E.: Efficient zero knowledge proofs of knowledge for homomorphisms. Ph.D. Thesis, Ruhr University Bochum (2005).
3. Bangerter E., Camenisch J., Maurer U.: Efficient proofs of knowledge of discrete logarithms and representations in groups with hidden order. In: Public Key Cryptography—PKC 2005. Lecture Notes in Computer Science, vol. 3386, pp. 154–171. Springer, Berlin (2005).
4. Bangerter E., Camenisch J., Krenn S.: Efficiency limitations for Σ-protocols for group homomorphisms. In: Theory of Cryptography TCC. Lecture Notes in Computer Science, vol. 5978, pp. 553–571. Springer, Berlin (2010).
5. Camenisch J., Kiayias A., Yung M.: On the portability of generalized Schnorr proofs. In: Advances in Cryptology—EUROCRYPT 2009. Lecture Notes in Computer Science, vol. 5479, pp. 425–442. Springer, Berlin (2009).
6. Cramer R.: Modular design of secure, yet practical cryptographic protocols. Ph.D. Thesis, University of Amsterdam (1996).
7. Damgård I.: On Σ-Protocols. Course Notes. Århus University, Denmark (2002).
8. Diffie W., Hellman M.E.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976).
9. Feige U., Fiat A., Shamir A.: Zero-knowledge proofs of identity. J. Cryptol. **1**, 77–94 (1988).
10. Fiat A., Shamir A.: How to prove yourself: practical solution to identification and signature problems. In: Advances in Cryptology—CRYPTO '86. Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer, Berlin (1987).
11. Goldwasser S., Micali S., Rackoff C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**, 186–208 (1989).
12. Guillou L.C., Quisquater J.-J.: A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In: Advances in Cryptology—EUROCRYPT '88. Lecture Notes in Computer Science, vol. 330, pp. 123–128. Springer, Berlin (1988).
13. Koyama K., Maurer U., Okamoto T., Vanstone S.A.: New public-key schemes based on elliptic curves over Z_n. In: Advances in Cryptology—CRYPTO '91. Lecture Notes in Computer Science, vol. 576, pp. 252–266. Springer, Berlin (1992).
14. Maurer U.: Unifying zero-knowledge proofs of knowledge. In: Preneel, B. (ed.) Advances in Cryptology—AFRICACRYPT 2009. Lecture Notes in Computer Science, vol. 5580, pp. 272–286. Springer, Berlin (2009).
15. Okamoto T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Advances in Cryptology—CRYPTO '92. Lecture Notes in Computer Science, vol. 740, pp. 31–53. Springer, Berlin (1992).
16. Pedersen T.: Non-interactive and information-theoretical secure verifiable secret sharing. In: Advances in Cryptology—CRYPTO '91. Lecture Notes in Computer Science, vol. 576, pp. 129–140. Springer, Berlin (1991).
17. Rivest R.L., Shamir A., Adleman L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978).
18. Schnorr C.P.: Efficient identification and signatures for smart cards. In: Advances in Cryptology—CRYPTO '89. Lecture Notes in Computer Science, vol. 435, pp. 239–252. Springer, Berlin (1990).