A Note on Security Proofs in the Generic Model

Marc Fischlin

Fachbereich Mathematik (AG 7.2)
Johann Wolfgang Goethe-Universität Frankfurt am Main
Postfach 111932
60054 Frankfurt/Main, Germany

marc@mi.informatik.uni-frankfurt.de
http://www.mi.informatik.uni-frankfurt.de/

Abstract. A discrete-logarithm algorithm is called generic if it does not exploit the specific representation of the cyclic group for which it is supposed to compute discrete logarithms. Such algorithms include the well-known Baby-Step-Giant-Step procedure as well as the Pohlig-Hellman algorithm. In particular, these algorithms match a lower bound of Nachaev showing that generic discrete-log algorithms require exponentially many group operations. Building on this lower bound, Shoup and subsequently Schnorr and Jakobsson proved other discrete-log-based protocols to be intractable in the generic model. Here, we discuss pitfalls when applying the generic model to other schemes than the discrete-log problem and when interpreting such lower bounds as security proofs for these schemes.

1 Introduction

The Baby-Step-Giant-Step algorithm and the Pohlig-Hellman algorithm to compute discrete logarithms in cyclic groups operate representation-independent, i.e., they do not rely on the specific representation of the group, and thus work for any cyclic group. These examples match a lower bound of Nachaev [20] proving that such generic algorithms need $\Omega(\sqrt{q})$ group operations to compute discrete logarithms in a group of size q. The index calculus method, though, defeats this lower bound as it requires subexponential time for groups like $\mathbb{Z}_p^*$ with standard binary encoding. Yet, the index calculus is not known to work for arbitrary groups, e.g., it seems to be inapplicable to elliptic curves [26].

From a theoretical point of view, it is easy to see that security proofs in the generic model do not generally transfer to "the real world" when adding an encoding, because generic algorithms might cause an exponential blow-up in comparison to Turing machines: for the group $(\mathbb{Z}_q, +)$ the discrete logarithm for some element $x \in \mathbb{Z}_q$ with respect to the generator 1 is simply x. If we use the standard binary encoding of $\mathbb{Z}_q$ then it is easy to compute discrete logarithms for an algorithm that operates on bit strings. A generic algorithm, however, requires $\Omega(\sqrt{q})$ steps to find the discrete logarithm, because it cannot take advantage of the trivial encoding.

Shoup [25] and subsequently Schnorr and Jakobsson [24] extended the idea of Nachaev and proved schemes relying on the discrete-logarithm problem to be intractable for generic algorithms. Nonetheless, applying this model and its lower bound to other discrete-log-based protocols should not be viewed as providing the same security level as the fact that, currently, optimal discrete-log finders for appropriate groups like elliptic curves are generic. The aim of this work is to highlight some of these pitfalls when proceeding from the discrete-log problem to more sophisticated schemes in the generic model.

We present a simple explanatory example. Moving from a computational task like computing discrete logarithms to a decisional problem, e.g., distinguishing encryptions, without additional consideration is a dangerous step: the representation in decisional problems is related to the problem at a different level. For instance, for discrete-log-based algorithms allegedly producing pseudorandom strings an inappropriate encoding of group elements may cause the output to be easily distinguishable from random, even though the encoding does not help to compute discrete logarithms.

As another example consider the signed ElGamal encryption scheme in [24]. Informally, a signed ElGamal encryption consists of an ordinary ElGamal encryption [11] together with a Schnorr signature [23]. In [24] it is shown that the signed ElGamal scheme is secure against adaptive chosen-ciphertext attacks [22] in a combination of the generic model and the random oracle model. This proof relies on the fact that the adversary cannot generate a group element without knowing a representation of this value with respect to a set of given group elements[1] and that this representation is known to a simulator reducing an adaptive attack with decryption requests to one without such queries.

In Section 3 we present a three-round negligible-error zero-knowledge protocol in the generic model for all languages in NP. Our protocol, too, applies the property that a generic adversary cannot compute group elements without being aware of a representation, and that a simulator knows these representations; for a similar but more complicated protocol see [16]. In [13] it has been proved that three-round negligible-error *black-box* (i.e., observing only external behavior of parties) zero-knowledge proofs can only exist for languages in BPP. Since our protocol does not obey this black-box approach —we see internal data of the simulated adversary, specifically, the representations of the group elements chosen by the adversary— we simplify the problem to achieve something which we do not know how to do otherwise. The same trick enables [24] to prove the signed ElGamal scheme to be unbreakable in this model.

Another problem with viewing intractability results in the generic model as security proofs is the dependency of cryptographic primitives in this setting. Consider the well-known Schnorr signature scheme [23] in which a signature corresponds to a proof of knowledge for the secret key. The challenge for this proof of knowledge is generated by applying a suitable hash function to a group

[1] A representation of X with respect to group elements $g_1, \ldots, g_n$ is a sequence $a_1, \ldots, a_n$ of integers such that $X = \prod g_i^{a_i}$. As for the special case $n = 1$ a representation corresponds to the discrete log of X with respect to g_1.

element and the message. This suggests that in the generic model the hash function itself must be treated as a black box, because it does not solely operate on bit strings but partially on group elements. The hash function is therefore closely connected to the group. As for an actual implementation in practice, it is therefore necessary that one verifies that choosing a good hash function to a presumably strong group does not give origin to undesired problems. We elaborate on this in Section 4 by discussing the Schnorr signature scheme. Namely, we show that, although this signature scheme seems to be secure in the generic model for appropriate hash functions, depending on the choice of the group to the hash function, in the real world we either obtain a secure combination or we get an easy-to-forge scheme.

In contrast to the generic approach, a classical proof in cryptography is modular: once proved secure under certain properties of the primitives involved, one can take *any* instantiation of *any* of the primitives satisfying these properties, and it is guaranteed that the combined scheme is secure. Hence, in the generic case additional attention must be paid when implementing the schemes. Since the aforementioned problems could be subtle and hidden very well, this introduces a dangerous source of flaws.

In conclusion, *even if some cryptographic protocol is provably secure in the generic model, this does not necessarily give us the same confidence as the observation that nowadays optimal algorithms for the discrete-logarithm problem for groups like elliptic curves are generic.*

2 Preliminaries

Since the aim of this work is to highlight principle problems with security proofs in the generic model, the following discussion and all results are kept at a very informal level.

Generic algorithms and their application to cryptography have been introduced to the crypto community by Shoup [25] relying on a result by Nachaev [20]. Shoup models generic algorithms by oracle Turing machines. That is, choose a random encoding of group elements and give the oracle Turing machine access to a group operation oracle taking as input the random encodings of two group elements X, Y and returning the random encoding of XY or XY^{-1}. Schnorr and Jakobsson [24] use the a slightly different approach by dividing data into group data and non-group data. “Their” generic algorithms operate on non-group data as Turing machines[2] whereas for group data the algorithm is only allowed to compute the group element $\prod X_i^{a_i}$ for elements $X_1, \dots, X_n$ and integers $a_1, \dots, a_n$ in a single oracle step.

In [24] the signed ElGamal encryption scheme —introduced in [17, 27]— has been analyzed in a combination of the random oracle and generic model. As mentioned before, basically, an encryption consists of an ElGamal encryption with a tag, a Schnorr signature. The system parameters are a group G of prime

[2] Actually, Schnorr and Jakobsson [24] allow the generic algorithms to compute arbitrary functions on the non-group data and do not even restrict to recursive functions.

order q, a generator g of G, and a random oracle $H : G^3 \to \mathbb{Z}_q$. The secret and the public key are given by $x \in \mathbb{Z}_q$ and $X = g^x$. In order to encrypt a message $m \in G$ select random $r, s \in \mathbb{Z}_q$, compute an ElGamal ciphertext $R = g^r, Y = mX^r$ and a Schnorr signature with g^s and $c = H(g^s, R, Y), z = s + cr \bmod q$. Finally, output (R, Y, c, z) as the ciphertext of m. To decrypt with the secret key x first check the validity of the signature tag, i.e., that $c = H(g^z R^{-c}, R, Y)$, and if so return the message Y/R^x. The idea is that the Schnorr signature with secret key r for message (R, Y) guarantees that the adversary knows r and thus the message $Y/X^r = m$.

A first formal proof that this combination of ElGamal encryption and Schnorr signature is indeed secure against adaptive chosen-ciphertext attacks has been given in [27], under the assumption that H is a random oracle, that the decisional Diffie-Hellman problem [6] is intractable, and based on a somewhat strong assumption about the unforgeability of Schnorr signatures. In [24] the scheme has been proved secure in the generic model given that H is a random oracle.

In order to show that certain approaches are possible when observing internal behavior, but are not known to be achievable otherwise, we present an example based on zero-knowledge proofs. Hence, we briefly discuss the definition of zero-knowledge proofs in the generic model. See [12] for a comprehensive treatment of zero-knowledge protocols. Informally, a zero-knowledge protocol [15] for a language $\mathcal{L}$ is an interactive proof system between an unbounded party, called the prover P, and a probabilistic polynomial-time machine, the verifier V, such that the following holds:

– completeness: if P and V both honestly follow the protocol then V always accepts inputs $x \in \mathcal{L}$.
– soundness: for $x \notin \mathcal{L}$ the verifier V only accepts with probability $\epsilon(|x|)$ for any malicious prover P^* pretending to be P. The function ϵ is called the error of the protocol. Likewise, if ϵ is negligible then the protocol has negligible error.
– zero-knowledge: for any $x \in \mathcal{L}$, any possibly malicious verifier V^* does not not learn anything useful beyond the fact that $x \in \mathcal{L}$ (in a computational sense) from the protocol execution with P. That is, for any verifier V^* there exists a probabilistic (expected) polynomial-time simulator S such that for $x \in \mathcal{L}$ the simulator's output $S(V^*, x)$ is computationally indistinguishable [14] from the random variable that describes the exchanged messages of a protocol execution between P and V^*.

Basically, augmenting interactive proof systems by a group oracle means to provide all parties P, P^*, V, V^*, S access to the same oracle. This, of course, implies that we have to transfer the instinguishability property to the generic model. Instead of demanding that any generic algorithm (with access to the same group oracle) cannot distinguish the prover's and the simulator's answers, we *afterwards* encode in both cases all group elements in a group with some encoding, like $\mathbb{Z}_p^*$ and the binary representation. Clearly, this leads to a conditional statement that the output of the zero-knowledge simulator is indistinguishable (in the standard sense) from the prover's answers *under the assumption that the group* *with the encoding is secure.* By this, we circumvent to introduce distinguishers in the generic model.

The zero-knowledge simulator which we present in Section 3 will not be a black-box simulator as it learns the queries of V^* submitted to the group oracle, and hence observes some internal behavior of V^*. In fact, this is crucial for our three-round negligible-error zero-knowledge protocol in the next section. Furthermore, this is exactly what is done in [24] in order to prove the signed ElGamal encryption scheme to be secure against adaptive chosen-ciphertext attacks. There, it is demonstrated that the adversary essentially cannot create ciphertexts without knowing the message, and that the message can be extracted by looking at the adversary's oracle gueries to the group oracle. Schnorr and Jakobsson [24] call this plaintext awareness (and thus implicitely suggest a definition in the generic model, although they do not present a formal definition). They refer to plaintext awareness as defined in [5] rather than to the refinement given in [2]. We are not aware if the signed ElGamal scheme is plaintext aware according to this refinement.

Finally, let us recall the three-round discrete-log-based oblivious transfer protocol of Bellare and Micali [3]. We apply this protocol as a tool to derive our zero-knowledge scheme. Informally, a chosen-one-out-of-two oblivious transfer scheme is a two-party protocol between a sender possessing messages m_0, m_1 and the receiver. The receiver would like to learn m_b from the sender such that the sender does not learn the receiver's choice b. On the other hand, the sender is willing to reveal one of the messages to the receiver, but does not want to give away anything about the other message. Bellare and Micali introduce the following protocol in a group G of prime order q generated by g.[3]

- The sender generates a random pair $x \in \mathbb{Z}_q$, $X = g^x$ of private and public key and sends X to the receiver.
- The receiver, trying to get m_b, randomly chooses $y \in \mathbb{Z}_q$, sets $Y_b = g^y$ and $Y_{b\oplus 1} = XY_b^{-1}$ and transmits Y_0, Y_1.
- The sender checks that $Y_0 Y_1 = X$. If so, it selects uniformly $a_0, a_1 \in \mathbb{Z}_q$ and computes the ElGamal encryptions $(A_i, B_i) = (g^{a_i}, Y_i^{a_i} m_i)$ for $i = 0, 1$ (where we presume for simplicity that the messages are in some way encoded as group elements). The sender transmits both pairs to the receiver.
- The receiver, knowing the discrete-log of Y_b, can decrypt m_b.

Intuitively, the receiver can only know one of the secret keys of Y_0, Y_1 and thus learns only a single message, i.e., the other message is computationally hidden under the decisional Diffie-Hellman assumption [3]. Conversely, the sender does not learn in an information-theoretical sense which of the messages the receiver

[3] In the generic model, the group is given, while in the real world it is generated by the sender in the first step, say, by selecting a subgroup G of $\mathbb{Z}_p^*$. Since only the sender's privacy but not the receiver's depends on the intractability of the discrete-log in this group, and because the receiver can verify that a proper group of prime order has been generated, we can simply assume that even a malicious sender chooses the group honestly.

wants to retrieve, because the values Y_0, Y_1 are distributed independently of b. We remark that the same functionality can be accomplished with a less efficient scheme based solely on the computational Diffie-Hellman assumption and hardcore predicates [3].

3 Three-Round Negligible-Error Zero-Knowledge for NP

In this section we present our three-round negligible-erro zero-knowledge protocol for the NP-complete language $\mathcal{DHC}$, Directed Hamiltonian Cycle. The well-known atomic protocol with error 1/2 works as follows (cf. [12]):

- The prover permutes the common input graph H with permutation π to obtain a graph $\pi(H)$. Then the prover sends a bit-wise commitment of the describing matrix of $\pi(H)$, and a commitment of π. We assume that the commitment scheme is non-interactive, computationally-hiding and unconditionally-binding; such commitment schemes exist for example under the discrete-log assumption (see [12]).
- The verifier chooses a random bit b and asks the prover to reveal a Hamiltonian cycle in the permuted graph ($b = 0$), or to show that the commitment sequence really contains an isomorphic copy of H ($b = 1$).
- The prover acts accordingly, i.e., for $b = 0$ opens n committed 1-bits of the matrix of $\pi(H)$ describing a directed Hamiltonian cycle, and for $b = 1$ decommits to all of π and $\pi(H)$.
- The verifier decides upon the opening.

Obviously, this protocol is complete. Soundness holds with error 1/2 because for input $H \notin \mathcal{DHC}$ the prover can answer at most one of the two possible challenges correctly. The zero-knowledge simulator tries to guess the challenge prior to the commitment, i.e., commits to an arbitrary graph with a random Hamiltonian cycle if the guess is $b = 0$, and to a random permutation of H for $b = 1$. Then it obtains the challenge b^* of the verifier V^* and if $b = b^*$ the simulator opens the commitments accordingly; else it restarts. The expected number of trials until the simulator successfully guesses b^* is two.

Assume that instead of opening the parts of commitment according to the challenge in the atomic protocol, the prover executes both openings in parallel but encrypts each of both sequences of decommitments with an independent secret key. Additionally, the prover transfers one of these keys obliviously to the verifier (who decides at random which key) with the Bellare-Micali protocol. This technique has already been successfully applied in other works about zero-knowledge proofs (e.g. [18, 9]). Completeness and soundness of the atomic protocol are preserved. But it is not clear that the zero-knowledge property still holds, because not knowing the right key the trial-and-error simulator above cannot check that its guess is correct. In the generic model, though, the simulator sees the verifier's group oracle queries and thus learns which key the verifier has chosen.

An important observation is that we can move the commitment step of the first round to the third round without affecting the soundness property. This leads to the protocol in Figure 3, where we use the notation $[\mathsf{OT}(K_0, K_1, b)]_i$ to denote the message in the i-th round ($i = 1, 2, 3$) of the OT scheme of Bellare-Micali with private input K_0, K_1 of the prover and random secret bit b of the verifier. Additionally, we denote by $\mathsf{Enc}_K(\cdot)$ a semantically-secure [14] encryption scheme, e.g., the basic ElGamal encryption scheme under the decisional Diffie-Hellman assumption [27], or a hardcore-predicate-based bit encryption scheme based on the computational Diffie-Hellman assumption.

prover P	common input: graph H	verifier V
	do $\lvert H\rvert$-times in parallel:	
select keys K_0, K_1		choose random b
	$\xrightarrow{[\mathsf{OT}(K_0, K_1, b)]_1}$	
	$\xleftarrow{[\mathsf{OT}(K_0, K_1, b)]_2}$	
choose random π compute $\pi(H)$ $C =$ bit-wise commitm. of $\pi, \pi(H)$ $E_0 = \mathsf{Enc}_{K_0}$(decomm. of HamCyc. in $\pi(H)$) $E_1 = \mathsf{Enc}_{K_1}$(decomm. of $\pi, \pi(H)$)		
	$\xrightarrow{C, E_0, E_1,\ [\mathsf{OT}(K_0, K_1, b)]_3}$	decrypt E_b check validity

Fig. 1. Three-Round Negligible-Error Zero-Knowledge Proof of Hamiltonian Cycle

Apparently, our protocol is complete. A malicious prover P^* can convince the verifier with probability at most $2^{-|H|}$ for inputs outside the language, because P^* must then lie either about the permutation or about the Hamiltonian cycle, but does not know on which side he is checked (the verifier's choice b is hidden information-theoretically in the Bellare-Micali protocol and the prover's commitments are unconditionally binding).

It remains to specify the zero-knowledge simulator S. In the generic model, S knows which key the verifier in each parallel execution retrieves, because the simulator sees the internal group oracle queries of the verifier. That is, although the malicious verifier might generate Y_0, Y_1 in the oblivous transfer protocol different than the honest verifier, it always holds in the generic model that $Y_0 = g^a X^b$ and $Y_1 = X \cdot Y_0^{-1}$ for some a, b which the simulator knows; thus, the verifier learns *at most* one of the keys K_0, K_1 and the simulator then knows which one (i.e., for $b \in \{0, 1\}$ the verifier knows the secret key to Y_b, and neither one for $b \notin \{0, 1\}$).

Our simulator imitates the simulator of the atomic protocol and chooses appropriate commitments and correct or dummy encryptions. That is, for each parallel execution our simulator selects keys K_0, K_1 and invokes in the oblivious transfer protocol with the verifier. The verifier answers with some group elements and we deduce which key (if any) the verifier wants to retrieve. If this is key K_0 we run the simulator of the atomic protocol to produce an appropriate commitment/decommitment for the guess $b = 0$; else, for K_1, we let this simulator generate a good instance for guess $b = 1$. We then correctly encrypt the opening with the chosen key and a produce dummy ciphertext of 0-bits with the other key. The fact that the views (afterwards encoded) are computationally indistinguishable under the decisional Diffie-Hellman assumption follows by standard techniques and is omitted. Also, if we use the less efficient variant of the Bellare-Micali protocol with hardcore predicates and the corresponding bit encryption scheme instead, then this scheme is computationally zero-knowledge under the computational Diffie-Hellman assumption.

Why does our result not contradict the lower bound in [13] for the round complexity of zero-knowledge proofs? The reason is that the BPP-algorithm in [13] relies on a black-box simulator observing merely the external behavior of the verifier. Here, the simulator sees some internal operations, namely, the queries to the group oracle. The same property has been used in [24]. Recall that a signed ElGamal encryption is a tuple $(R, Y, c, z) = (g^r, mX^r, c, z)$. In [24] it has been shown that submitting such a tuple to the decryption oracle can be simulated because the signature ensures that the adversary must have computed g^r via the group oracle with very high probability; thus, r is known to a simulator *that keeps track of the adversary's group oracle queries*. In particular, it follows that the answer $m = Y/X^r$ of the decryption oracle can be computed by the simulator without knowing the secret key to X. By this, any decryption requests can be simulated by a single group operation.

4 Instantiations of the Schnorr Signature Scheme

In this section we discuss problems when choosing bad combinations of instantiations of the primitives, although the constructed scheme is presumably secure in the generic model. Our demonstration example will be the Schnorr signature scheme [23]. First, let us briefly recall the system.

The scheme involves a group G of prime order q generated by some $g \in G$ and a hash function H; we will later discuss the properties of H. The secret key is a random element $x \in \mathbb{Z}_q$ and the public key is given by $X = g^x \in G$. To sign a message m the signer chooses a random $r \in \mathbb{Z}_q$, computes g^r and $c = H(g^r, m) \in \mathbb{Z}_q$ as well as $y = r + cx \bmod q$. The signature consists of the pair (c, y). In order to verify a signature/message pair $(c, y), m$ the verifier calculates $Z = g^y X^{-c}$ and checks that $c = H(Z, m)$.

A potential attack on the Schnorr signature scheme is to reverse engineer the hash function, i.e., to choose a hash value c beforehand and then to try to find $y \in \mathbb{Z}_q$ such that $H(g^y X^{-c}, m) = c$. Obviously, (c, y) is then a valid signature. In practice, it is therefore assumed that H is a collision-intractable hash function that cannot be reverse engineered.

We add the public parameters describing the group G and g, X to the hash evaluation process. That is, the hash value is computed as $H(\langle G \rangle, g, X, g^r, m)$ and as $H(\langle G \rangle, g, X, Z, m)$, respectively, where $\langle G \rangle$ denotes the group description. This is also suggested in [19] to prevent so-called adversarial hashing, and to best of our knowledge this does not weaken the Schnorr signature scheme. Nonetheless, it gives us the possibility to relate the hash function evaluation process to the underlying group.

We will consider two instantiations of the collision-intractable hash function and the group. Both instances use the same hash function, but each time a different cryptographically-strong group. One example will be completely insecure, whereas the other seems to be provide a secure signature scheme. By this, it follows that the choice of the group also affects the choice of the hash function and vice versa. As we will argue, both approaches conceivably provide a secure combination in the generic model. In contrast, a traditional security proof that a safe group and a collision-intractable hash function withstanding reverse engineering are sufficient would imply that any combination of, say, SHA-1 or MD5 with groups in $\mathbb{Z}_p^*$ or elliptic curves yields a secure scheme. Hence, security in the generic model does not support modular implementations in general.

For sake of clarity, we explain the example below for subgroups of $\mathbb{Z}_p^*$ of prime order q with binary encoding. It also works for any other group, say, elliptic curves, if we hash down the binary representations of group elements to numbers between 0 and $p-1$. Let h be a collision-intractable hash function that maps bit strings to the intervall $[1, (q-1)/2]$, viewed as a subset of $\mathbb{Z}_q$. Furthermore, let h be secure against reverse engineering in the sense discussed above. Define the hash function H for the signature scheme by dividing the input message m into m_1, m_2 where $m_2 \in \{0,1\}^{|p|}$ is interpreted as a group element in $\mathbb{Z}_p^*$. Set

$$\begin{aligned} &H(p, q, g, X, R, m_1, m_2) \\ &\quad = \begin{cases} h(m_1) & \text{if } R \in [0, q) \text{ and } RX^{h(m_1)} = g \bmod p \\ & \quad \text{and } g^R = m_2 \bmod p \\ h(R, m_1, m_2) + \frac{q-1}{2} \bmod q & \text{else} \end{cases} \end{aligned}$$

It is easy to see that he derived hash function H is collision-intractable for fixed p, q, g, X and varying (R, m_1, m_2).

The idea of the construction of H is that its properties depend on the group. Specifically, assume that we choose $p = 2q+1$. Then roughly half of the elements in $G \subseteq \mathbb{Z}_p^*$ fall into the intervall $[0, q)$ (see [21]). If an adversary now picks m_1 at random and computes $R = gX^{-h(m_1)}$ then with probability approximately $1/2$ this value R is less than q as a natural number (assuming that the hash function h distributes random values quite well). In this case, $RX^{h(m_1)} = g$ and for $m_2 = g^R \bmod p$ the hash funtion output equals $c = h(m_1)$. Thus, $(c, 1)$ is a valid signature for (m_1, m_2) and the adversary easily succeeds in forging Schnorr signatures (without necessarily being able to compute discrete logarithms).

Now let $p = wq + 1$ for $w \gg q^2$. Assume for the moment that except for 1 none of the other $q - 1$ group elements lies in $[0, q]$. Unfortunately, we do not know whether this holds in general or not, and we are not aware of any results about the distribution of elements of this subgroup in $\mathbb{Z}_p^*$ (if the elements are almost uniformly in $\mathbb{Z}_p^*$ then this clearly follows from the choice of w). But again, we stress that this is an instructive example and we therefore admit this simplification. In this case, the hash function evaluation for (R, m_1, m_2) can only result in $h(m_1)$ if $R = 1$. But then $RX^{h(m_1)} = X^{h(m_1)} = g$ is only possible for $\log_g X = 1/h(m_1) \bmod q$. This, in turn, is equivalent to computing the discrete-logarithm of X to base g, and assuming the intractability of the discrete-log problem it is therefore very unlikely that this happens. Hence, given that the case above never occurs, the hash function evaluation merely results in values $h(R, m_1, m_2) + (q-1)/2 \bmod q$ and the scheme resembles to the original Schnorr system and is thus believed to be secure.

What happens in the generic model of Schnorr-Jakobsson? There, the adversary cannot interchange group data and non-group data. Hence, any hash function query cannot yield the answer $h(m_1)$ and the scheme is again conceivably secure. In other words, due to the generic model the hash function H has the additional property of being immune against reverse engineering, although H has not for the wrong choice of the group when implementing.

5 Conclusion

We have pointed out several pitfalls for security proofs in the generic model. Clearly, it is preferable to construct attractive protocols that are provably secure by classical methods. Yet, for some schemes used in practice like DSS such security proofs are still missing today (assuming that DSS can be proven secure at all). It is therefore a worthwhile effort to consider certain attacks on these schemes. But one should have in mind that it merely provides some evidence of hardness if these attacks fail. Also, the lack of proofs should incite researchers to find provably secure alternatives.

We remark that there are alternatives to the signed ElGamal scheme of [17, 27, 24] which are also discrete-log-based but require milder, yet still "non-standard" assumptions. One is the system based on the random oracle assumption and arbitrary trapdoor functions [5]. Another one is the DHAES scheme of Abdalla et al. [1] based on a potentially stronger assumption than the decisional Diffie-Hellmann assumption. The DHAES scheme seems to be at least as efficient as the signed ElGamal scheme: one exponentiation is traded for some private-key operations.

Finally, we remark that there is the ingenious encryption scheme of Cramer and Shoup [8] based only on the decisional Diffie-Hellman assumption; the

Cramer-Shoup scheme is only slightly less efficient than the signed ElGamal scheme.

Acknowledgements

We thank Claus Schnorr for remarks on the generic model.

References

1. M.Abdalla, M.Bellare, P.Rogaway: DHAES: An Encryption Scheme Based on the Diffie-Hellmann Problem, *available at* `http://www-cse.ucsd.edu/users/mihir/`, 1998.
2. M.Bellare, A.Desai, D.Pointcheval, P.Rogaway: Relations Among Notions of Security for Public-Key Encryption Schemes, *Crypto '98, Lecture Notes in Computer Science, Vol. 1462, Springer-Verlag, pp. 26–45*, 1998.
3. M.Bellare, S.Micali: Non-Interactive Oblivious Transfer and Applications, *Crypto '89, Lecture Notes in Computer Science, Vol. 435, Springer-Verlag, pp. 547–559*, 1990.
4. M.Bellare, P.Rogaway: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, *ACM Conference on Computer and Communication Security, pp. 62–73*, 1993.
5. M.Bellare, P.Rogaway: Optimal Assymetric Encryption, *Eurocrypt '94, Lecture Notes in Computer Science, Vol. 950, Springer-Verlag, pp. 92–111*, 1994.
6. D.Boneh: The Decision Diffie-Hellman Problem, *Third Algorithmic Number Theory Symposium, Lecture Notes in Computer Science, Vol. 1423, Springer-Verlag, pp. 48–63*, 1998.
7. R.Canetti, O.Goldreich, S.Halevi: The Random Oracle Methodology, Revisited, *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing, pp. 209–218*, 1998.
8. R.Cramer, V.Shoup: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attacks, *Crypto '98, Lecture Notes in Computer Science, Vol. 1462, Springer-Verlag, pp. 13–25*, 1998.
9. A.De Santis, G.Di Crescenzo, G.Persiano: Public-Key Cryptography and Zero-Knowledge Arguments, *Information and Computation, Vol. 121, No. 1, pp. 23–40*, 1995.
10. W.Diffie, M.Hellman: New Directions in Cryptography, *IEEE Transaction on Information Theory, Vol. 22, pp. 644–654*, 1976.
11. T.ElGamal: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, *IEEE Transaction on Information Theory, Vol. 31, pp. 469–472*, 1985.
12. O.Goldreich: Foundations of Cryptography (Fragments of a Book), *available at* `http://www.wisdom.weizmann.ac.il/home/oded/public_html/index.html`, 1998.
13. O.Goldreich, H.Krawczyk: On the Composition of Zero-Knowledge Proof Systems, *SIAM Journal on Computing, Vol. 25, pp. 169–192*, 1996.
14. S.Goldwasser, S.Micali: Probabilistic Encryption, *Journal of Computer and System Sciences, Vol. 28(2), pp. 270–299*, 1984.

15. S.Goldwassser, S.Micali, C.Rackoff: The Knowledge Complexity of Interactive Proof Systems, *SIAM Journal on Computing, Vol. 18, pp. 186–208*, 1989.
16. S.Hada, T.Tanaka: On the Existence of 3-Round Zero-Knowledge Protocols, *Crypto '98, Lecture Notes in Computer Science, Vol. 1462, Springer-Verlag, pp. 408–423*, 1998.
17. M.Jakobsson: A Practical Mix, *Eurocrypt '98, Lecture Notes in Computer Science, Vol. 1403, Springer-Verlag, pp. 448–461*, 1998.
18. J.Kilian, S,Micali, R.Ostrovsky: Minimum Resource Zero-Knowledge Proofs, *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, 1989.
19. S.Micali, L.Reyzin: Signing with Partially Adversarial Hashing, *available at* `http://theory.lcs.mit.edu/~reyzin`, 2000.
20. V.Nechaev: Complexity of a Determinate Algorithm for the Discrete Logarithm, *Mathematical Notes, Vol. 55, pp. 165–172*, 1994.
21. R.Peralta: On the Distribution of Quadratic Residues and Non-Residues Modulo a Prime Number, *Mathematical Notes, Vol. 58*, 1995.
22. C.Rackoff, D.Simon: Non-Interactive Zero-Knowledge Proofs of Knowledge and Chosen Ciphertext Attacks, *Crypto '92, Lecture Notes in Computer Science, Vol. 576, Springer-Verlag, pp. 433–444*, 1998.
23. C.Schnorr: Efficient Signature Generation for Smart Cards, *Journal of Cryptology, Vol. 4, pp. 161–174*, 1991.
24. C.Schnorr, M.Jakobsson: Security of Signed ElGamal Encryption, *Asiacrypt 2000, Lecture Notes in Computer Science, Springer-Verlag*, 2000.
25. V.Shoup: Lower Bounds for Discrete-Logarithm and Related Problems, *Eurocrypt '97, Lecture Notes in Computer Science, Vol. 1233, Springer-Verlag, pp. 256–266*, 1997.
26. J.Silverman, J.Suzuki: Elliptic Curve Discrete Logarithms and the Index Calculus, *Asiacrypt '98, Lecture Notes in Computer Science, Vol. 1514, Springer-Verlag, pp. 110–125*, 1998.
27. Y.Tsiounis, M.Yung: On the Security of ElGamal Based Encryption, *PKC '98, Lecture Notes in Computer Science, Vol. 1431, Springer-Verlag, pp. 117–134*, 1998.