One-wayness of key generation function of a Pairing-Based Identification Protocol

December 2, 2021

Abstract

In previous work, we have proven various security properties of a pairing-based identification protocol. We used a chain of four smaller proofs, which then together resulted in a full security argument. However, one these smaller proofs was indeed not proven: It was merely claimed that "It is obvious that the computation (...) is one-way under the DL assumption.", in the false believe, this reduction is indeed straight forward. This is unfortunately not the case, and this report is an attempt to correct this oversight.

1 Introduction

We refer for a full motivation and introduction of the pairing-based protocol to our previous report. We only repeat here what directly applies to the key generation function, as this is the only part of the protocol under concern in this report.

The idea behind the protocol, and therefore also behind the key generation function, is to allow as short secrets as possible. As these secrets have to be typed in by the user in the foreseen use case (internet voting), having short secrets improves usability. The proposed protocol is in this sense optimal; with the length of the secret equaling its security parameter. Notably, the secret size does not have to be doubled as DL-breaking algorithms such as polland-rho cannot be employed to find the secret directly.

Contributions We provide three proof attempts to show the key generation function is one-way.

2 Cryptographic Primitives

We clarify notation, terminology and cryptographic primitives we use throughout this work.

Notation We use upper-case letters for sets and lower-case letters for their elements (like $X = \{x_1, x_2, ..., x_n\}$). $|X|$ denotes the cardinality of the set X.

For algorithms, we use bold letters (like **Sign**). We denote $\overset{\$}{\leftarrow}$ when we choose an element uniform at random (like $x \overset{\$}{\leftarrow} X$). We forgo modeling the source of randomness explicitly; we simply assume each algorithm has access to randomness of suitable size.

Terminology A *public key* can be publicly known, and allows to verify a signature (or to encrypt a message). The corresponding *private key* must only be known to the signer (or decryptor).

Groups A (multiplicative) group $\Gamma = (G, *, ^{-1}, 1)$ is an algebraic structure consisting of a set G of elements, the binary operation $* : G \times G \to G$, the unary operation $^{-1} : G \to G$, and the neutral element 1. For any $\{x, y, z\} \subset G$, the following holds: Associativity ($(x * y) * z = x * (y * z)$), identity element ($1 * x = x * 1 = x$), and inverse element ($x * x^{-1} = 1$). x^k applies the group operator $k - 1$ times to x. An element $g \in G$ is called a generator of G, iff $\{g^1, ..., g^p\} = G$ for $p = |G|$. If the group is of prime order, every element is a generator, except the neutral element 1. After this paragraph, our notation will refer to Γ using G.

With $\mathbb{Z}_p^*$ we denote the multiplicative group of integers in which multiplications are computed modulo the prime p. With $\mathbb{Z}_p$ we denote the additive group of integers in which additions are computed modulo p. We handle negative values as $-k * x = k * (-x) = -(k * x)$ and $x^{-k} = (x^{-1})^k = (x^k)^{-1}$.

For p prime, we can define the prime-order field $(\mathbb{Z}_p, +, *, -, ^{-1}, 0, 1)$ combining the additive group $(\mathbb{Z}_p, +, -, 0)$ and the multiplicative group $(\mathbb{Z}_p^*, *, ^{-1}, 1)$ into a single algebraic structure with the additional property of distributivity of multiplication over addition ($(x+y)*z = (x*z)+(y*z)$ for any $\{x, y, z\} \subset \mathbb{Z}_p$).

Pairing A map $\theta : X \times Y \to Z$ is called a pairing if it provides

- *bilinearity* ($\theta(x_1 * x_2, y) = \theta(x_1, y) * \theta(x_2, y)$ and $\theta(x, y_1 * y_2) = \theta(x, y_1) * \theta(x, y_2)$),
- *non-degeneracy* (for all generators x and y, $\theta(x, y)$ generates Z) and
- *efficiency* (θ is efficiently computable).

We call a pairing *Type 1* if $G_1 = G_2$, *Type 2* if $G_1 \neq G_2$ but there exists a homomorphism from G_2 to G_1, and *Type 3* if $G_1 \neq G_2$ and there exists no homomorphism. [GPS08]. Implementation of pairings are feasible using the Tate or the Weil pairing, applying Miller's algorithm [Lyn07].

2.1 Cryptographic basics

Negligible A function $\epsilon : N \to [0,1]$ is *negligible* if for all $c \geq 0$ there exists $k_c \geq 0$ such that $\epsilon(k) \leq \frac{1}{k^c}$ for all $k > k_c$ [Kat10]. We declare a cryptographic scheme as *secure* if the success probability of the attacker to reach its goal using its assigned capabilities is negligible.

Security parameter A *security parameter* describes the cryptographic security of a scheme; the amount of computational power required to break a scheme or property by a polynomially bounded adversary [Kat10]. We denote the security parameter as 1^k.

Hardness Assumptions The Discrete Log (DL) assumption states, that it is hard to find x for given $y = g^x$. The computational Diffie-Hellman (CDH) assumption states, that it is hard to compute g^{ab} from given $y = g^a$ and $z = g^b$. Further, the decisional Diffie-Hellman (DDH) states it is hard to differentiate $(g^a,\ g^b,\ g^{ab})$ and $(g^a,\ g^b,\ g^c)$. [Kat10] One can easily verify that solving DL implies solving CDH, and solving CDH implies solving DDH (while the reverse does not hold). It is assumed that DDH (and hence CDH and DL) holds in $G \subset \mathbb{Z}_p^*$ for $|G|$ prime [HKLD17].

Solving the discrete log Some algorithms solving the discrete log exist which are faster than simply bruteforcing. For p the prime group order, the best algorithms available (2015) are variants of the deterministic baby-step giant-step algorithm (succeeds in $O(\sqrt{p})$ time and space) or the probabilistic Pollard's rho (low space, $O(\sqrt{p})$ time) [GWZ].

3 KeyGen of the Pairing Based Identification Protocol

We now present the KeyGen function of the Pairing Based Identification Protocol in 1. Publicly known is the security parameter 2^k, the cyclic groups G_1 and G_2 of order $p \geq 2k$ with its generators $g_1 \in G_1$ and $g_2 \in G_2$, and the pairing $\theta : G_1 \times G_2 \to G_T$.

Algorithm 1 KeyGen of the Pairing Based Identification Protocol

```
$(r, x) \xleftarrow{\$} (\mathbb{Z}_p \times \mathbb{Z}_{2^k})$
$(y_1, y_2) \leftarrow (g_1^r, g_2^{r+x})$
return $(sk = x, pk = (y_1, y_2))$.
```

We claim KeyGen to be one-way under some variant of the discrete log assumption. We further claim that x can directly equal the security parameter, when we choose r at least double the security parameter, as the runtime of the DL-breaking algorithms depend on r.

3.1 One-way KeyGen proof proposals

We first define the one-way KeyGen game. Then we present proof drafts with each independently establishing one-wayness. Each draft however has its own issues, which is clearly marked using colors: Orange for issues which make the proof less credible, and red for issues that make the proof unsound.

Game 1 (one-way KeyGen)**.** Let KeyGen be a key generation algorithm for $R \subset X \times Y$. For a given adversary A, the attack game runs as follows:

- The challenger runs $(sk, pk) \xleftarrow{\$} \mathit{KeyGen}()$, and sends $pk = y$ to A.
- A outputs $\tilde{x} \in X$.

We say that A wins the game if $(\tilde{x}, y) \in R$. If the probability is negligible that A wins the game, KeyGen is one-way [BS17, Attack Game 19.2].

3.1.1 Proof Proposal 1

The proof reduces to the *Discrete Logarithm with Short Exponents* (c-DLSE) assumption. It then argues for the preconditions to enable short key size.

Definition 1 (c-DLSE assumption)**.** The Discrete Logarithm with Short c-Bit Exponents (c-DLSE) assumption states that it is hard to find x for given $y = g^x$ with $x \leq 2^c$. For n the group size, the best known algorithm to compute discrete logarithms is the index calculus method which runs in time sub-exponential in n. If we set $c = \log n$[1], there are no known poly-time algorithms that can compute

[1]The original papers state $c = \omega(\log n)$ but do not define this ω [PS98, Gen00].

x. Given c, we get $\frac{c}{2}$ bits of security as the runtime of the baby-step giant step and pollard-rho algorithms are in $O(2^{\frac{c}{2}})$. The assumption has been introduced to proof PRGs secure [PS98, Gen00, V$^+$13]. The c-DSLE assumption is weaker than the discrete log assumption [V$^+$13].

Reduction 1 (c-DLSE $\Rightarrow$ one-way KeyGen)**.** Our c-DLSE challenger gives us $y = g_2^x$ with $x \leq 2^c$, for c the keysize of the secret key. We transform this into $pk = (g_1^r, g_2^r * y)$ given $r \overset{\$}{\leftarrow} \mathbb{Z}_p$. Note that this public key is well-formed.[2] We send this to our KeyGen adversary, which returns x. We return x to our challenger, and win the game at least when the KeyGen adversary wins. Note that the KeyGen operates on much bigger values than given by the challenger; hence solves a much more "difficult" problem than we were challenged to do so.

DL in G_1 and independent g_1, g_2 to reduce key size The preceding reduction only requires discrete logarithm in G_2, but the key size must be at least double the security parameter (by the c-DLSE assumption). We observe however that our KeyGen never publishes g_2^x directly, but only g_2^{r+x} instead. If finding g_2^r is hard, then g_2^{x+r} effectively hides x, allowing us to reduce the size of x to equal the security parameter. Concretely, if we choose r at least double the security parameter 1^k, the efficient algorithms then run in at least $O(\frac{2*1^k}{2})$, hence at least in $O(1^k)$.

There are two ways how to find g_2^r. We can transform g_1^r into g_2^r if the logarithm between the two generators is known, hence we require our pairing to use independent generators g_1 and g_2.[3] Further, we can extract r directly out of g_1^r, which motivates our DL assumption on G_1. This is not a proof. These two requirements are necessary, but are they really sufficient?

3.1.2 Proof Proposal 2 [unsound]

The proof reduces to the *Discrete Logarithm* assumption adapted to two groups (co-DL). For this to work, it modifies the KeyGen function to return r as part of the secret key, and argues this modification sound.

Definition 2 (co-CDH assumption)**.** The Computational Discrete Logarithm assumption adapted to two groups states that it is hard to find $g_1^{\alpha\beta}$ for given $g_1^\alpha, g_1^\beta, g_2^\alpha$. Note that for symmetric pairings (where $g_1 = g_2$), the assumption is identical to CDH. The assumption has been introduced to proof BLS-signatures secure [BS17].

Definition 3 (co-DL assumption)**.** The Discrete Logarithm assumption adapted to two groups states that it is hard to find α for given g_1^α, g_2^α. Note that for symmetric pairings (where $g_1 = g_2$), the assumption is identical to DL. This assumption is proposed for the first time. Motivating it is straightforward from the co-CDH assumption; but why was it never proposed before? Does some property of the pairing make this nonsensical?

[2]Its distribution is equal to public keys generated by KeyGen.
[3]Note that this requires Type 3 pairings.

Modified KeyGen function We modify the KeyGen function to include the variable r into the private key. Note that r is never used throughout the execution of the protocol (neither by the prover nor by the verifier).

Algorithm 2 KeyGen' of the Pairing Based Identification Protocol which returns additionally r as part of the private key.

```
$(r, x) \overset{\$}{\leftarrow} (\mathbb{Z}_p \times \mathbb{Z}_{2^k})$
$(y_1, y_2) \leftarrow (g_1^r, g_2^{r+x})$
return $(sk = (x, r), pk = (y_1, y_2))$.
```

This transformation is likely invalid, as it cannot be proven that KeyGen' $\leq$ KeyGen (in fact, KeyGen' $\geq$ KeyGen). As a simple example why this construction is invalid, let G be an invertible key generation function. Output additionally x in the secret key, and g^x in the public key, and call this construction G'. G' can now be proven one-way given the logic presented in this subsection, although clearly the protocol using keys of G is still insecure.

Reduction 2 (co-DL $\Rightarrow$ one-way KeyGen)**.** Our co-DL challenger gives us $y_1 = g_1^\alpha, y_2 = g_2^\alpha$. We transform this into $pk = (y_1, y_2 * g_2^x)$ given $x \overset{\$}{\leftarrow} \mathbb{Z}_{2^k}$. Note that this public key is well-formed.[4] We send this to our KeyGen adversary, which returns (x, r). We return r to our challenger, and win the game at least when the KeyGen adversary wins (exactly when $r = \alpha$).

3.1.3 Proof Proposal 3

The proof reduces to the *Bilinear Discrete Logarithm with Short c-Bit Offset in Exponent* (c-BDLSOE) assumption. We clearly define this assumption and show it weaker than DL. Then we use it to proof KeyGen one-way.

Definition 4 (c-BDLSOE assumption)**.** The Bilinear Discrete Logarithm with Short c-Bit Offset in Exponent (c-BDLSOE) assumption states that it is hard to find β for given $y_1 = g^\alpha, y_2 = g_2^{\alpha+\beta}$ with $\beta \leq 2^c$. For n the group size, the best known algorithm to compute discrete logarithms is the index calculus method which runs in time sub-exponential in n. If we set $c \leq \frac{\log n}{2}$, there are no known poly-time algorithms that can compute β. Given c, we get c bits of security as the runtime of the baby-step giant step and pollard-rho algorithms are in the length of $O(2^{\frac{n}{2}} \geq 2^c)$. The c-BDLSOE assumption is weaker than the discrete log assumption in both G_1 and G_2 (we present a reduction). Introducing a new assumption is far from optimal, even if it can be showed that it is weaker than DL. Further, claiming "no known algorithms exist" for a newly introduced assumption is a weak argument.

Reduction 3 (c-BDLSOE $\leq$ DL G_1 and DL G_2)**.** Our c-BDLSOE challenger gives us $y_1 = g^\alpha, y_2 = g_2^{\alpha+\beta}$ with $\beta \leq 2^c$. We query the DL G_1 adversary with

Feedback: Reduction is wrong

[4]Its distribution is equal to public keys generated by KeyGen.

y_1 to get x_1 back. We then query the DL G_2 adversary with y_2 to get x_2 back. We return $x_2 - x_1$, and win the game at least when both adversaries of DL G_1 and DL G_2 succeeded.

Reduction 4 (cBDLSOE $\Rightarrow$ one-way KeyGen)**.** Our c-BDLSOE challenger gives us $y_1 = g^{\alpha}, y_2 = g_2^{\alpha+\beta}$ with $\beta \leq 2^c$. We query the one-way KeyGen adversary with (y_1, y_2) and get back x. We return this x, and win the game at least when the one-way adversary succeeded.

References

[BS17] Dan Boneh and Victor Shoup. A graduate course in applied cryptography. *Retrieved from https://crypto.stanford.edu/˜dabo/cryptobook/BonehShoup_0_5.pdf*, 2017.

[Gen00] Rosario Gennaro. An improved pseudo-random generator based on discrete log. In *Annual International Cryptology Conference*, pages 469–481. Springer, 2000.

[GPS08] Steven D Galbraith, Kenneth G Paterson, and Nigel P Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[GWZ] Steven D Galbraith, Ping Wang, and Fangguo Zhang. Computing elliptic curve discrete logarithms with improved baby-step giant-step algorithm. *Advances in Mathematics of Communications*, 11(3):453.

[HKLD17] Rolf Haenni, Reto E Koenig, Philipp Locher, and Eric Dubuis. Chvote system specification. *IACR Cryptol. ePrint Arch.*, 2017:325, 2017.

[Kat10] Jonathan Katz. *Digital signatures.* Springer Science & Business Media, 2010.

[Lyn07] Ben Lynn. *On the implementation of pairing-based cryptosystems.* PhD thesis, Stanford University Stanford, California, 2007.

[PS98] Sarvar Patel and Ganapathy S Sundaram. An efficient discrete log pseudo random generator. In *Annual International Cryptology Conference*, pages 304–317. Springer, 1998.

[V$^+$13] F Vercauteren et al. Final report on main computational assumptions in cryptography. *European Network of Excellence in Cryptography II*, 11, 2013.