# T-79.512 Cryptology: Special Topics / Pairing-based Cryptography: Short Signatures from Bilinear Pairings

Johan Wallén

Draft, January 30, 2003

## 1 Introduction

In several applications it is important to have very short digital signatures. This scenario arises for example when the user have to type in a signature by hand and in low-bandwidth communications environments. In standard ElGamal [6] style signature schemes such as DSA [10], the signature consists of two group elements. In applications where these signatures are too long, it is natural to ask for a discrete logarithm or Diffie-Hellman based signature scheme where the signature consists of only a single group element. Such a scheme was first proposed in [3]. Their signature scheme works in any group $G$ where the computational Diffie-Hellman (CDH) problem is hard but the decisional Diffie-Hellman (DDH) problem is easy. Such groups are called *gap Diffie-Hellman groups*. If $G$ is of prime order, the CDH problem is hard in $G$, and there exists an efficiently computable bilinear pairing $\hat{e}\colon G^2 \to H$ for some group $H$ with $|H| = |G|$, it is easy to see that $G$ is a gap Diffie-Hellman group. In fact, all popular candidates for gap Diffie-Hellman groups like super-singular elliptic curves arise in this way. We refer to the other surveys in this seminar for a background on bilinear pairings.

Initially, bilinear pairings where used in *destructive* ways by showing that the DDH problem is easy in certain groups [9]. The signature scheme [3] was together with [7, 11, 2] among the first examples of *constructive* use of pairings by presenting improved key exchange protocols and encryption schemes based on pairings.

The gap signature scheme from [3] is simple and has a very efficient signing algorithm. Signing a message involves only a single hash function evaluation and a group exponentiation. Verification is more complicated, since it involves solving one decisional Diffie-Hellman instance.

The scheme is secure against existential forgery under a chosen message attack in the random oracle model [1]. In this model, both legitimate users and the attacker are assumed to have oracle access to a random function. This is usually used to model hash functions. Since the random oracle model is in general unsound [4], the security proof unfortunately only shows heuristically the security of the scheme. The proof is conceptually very simple and gives a straightforward reduction of the computational Diffie-Hellman problem to existentially forging the signature scheme.

In this survey, we will present the gap signature scheme from [3]. We start with a short discussion of gap Diffie-Hellman groups. After that, we present the signature scheme and the define security against existential forgery under chosen ciphertext attacks. Finally, we present a security proof in the random oracle model.

## 2 Gap Diffie-Hellman groups

Let $G$ be a cyclic (multiplicative) group of prime order, $|G| = p$. We consider two problems in $G$. The computational Diffie-Hellman (CDH) problem is given $(g, g^a, g^b)$ to compute $g^{ab}$. The decisional Diffie-Hellman (DDH) problem is to distinguish between the distributions $(g, g^a, g^b, g^{ab})$ and $(g, g^a, g^b, g^c)$, where $a, b, c \in \mathbb{Z}_p^*$ are uniformly distributed. Tuples of the form $(g, g^a, g^b, g^{ab})$ are called valid Diffie-Hellman tuples. The group $G$ is a gap Diffie-Hellman (GDH) group if DDH is easy but CDH is hard in $G$.

To make this definition more precise, we fix a model of computation such as probabilistic RAM machines. We assume that the model is normalised so that one group operation can be carried out in unit time.

**Definition 1.** The group $G$ is a $\tau$-decision group for Diffie-Hellman, if the DDH problem in $G$ can be solved in time $\tau$.

We let $x \leftarrow A$ denote that $x$ is chosen from $A$ according to a probability distribution (hopefully) understood from context.

**Definition 2.** The advantage of an algorithm $A$ in solving the CDH problem in a group $G$ of prime order $p$ is

$$\mathrm{adv}_A^G = \Pr[a, b \leftarrow \mathbb{Z}_p^* : A(g, g^a, g^b) = g^{ab}] \ ,$$

where the probability is taken over the choice of $(a, b)$ and the internal coin tosses of $A$. The algorithm $A$ is said to $(t, \epsilon)$-break CDH in $G$ if $A$ runs in time at most $t$ and $\mathrm{adv}_A^G \geq \epsilon$.

**Definition 3.** A group $G$ of prime order is a $(\tau, t, \epsilon)$-gap Diffie-Hellman group if it is a $\tau$-decision group for Diffie-Hellman and no algorithm $(t, \epsilon)$-breaks computational Diffie-Hellman in $G$.

The most popular candidates for GDH groups are certain classes of super-singular elliptic curves. In these groups, the DDH problem can be solved using the Weil pairing [12]. Suggestions for selecting GDH groups with suitable parameters can be found in [3]. A concrete example of a group where the DDH problem is easy but the CDH problem is equivalent to a (presumably hard) discrete logarithm problem is given in [8].

# 3 The gap signature scheme

Fix a GDH group $G$ of prime order $p$ and denote $G^* = G - \{1\}$. Fix an element $g \in G^*$. The signature scheme will use a full domain hash function $h\colon \{0,1\}^* \to G^*$ [1, 5]. In the security analysis, we will model $h$ as a uniformly selected function $\{0,1\}^* \to G^*$. The gap signature scheme consists of three algorithms, keyGen, sign and verify.

**Key generation.** Pick a random $x \in \mathbb{Z}_p^*$ and compute $v = g^x$. The public key is $v$ and the secret key is $x$.

**Signing.** Given a secret key $x$ and a message $M \in \{0,1\}^*$, compute $m = h(M)$ and $\sigma = m^x$. The signature is $\sigma \in G^*$.

**Verification.** Given a public key $v$, a message $M$ and a signature $\sigma$, compute $m = h(M)$ and verify that $(g, v, m, \sigma)$ is a valid Diffie-Hellman tuple.

If the signature is correctly computed, $m = h(M) = g^a$ for some $a \in \mathbb{Z}_p^*$ and thus $(g, v, m, \sigma) = (g, g^x, g^a, g^{ax})$ indeed is a valid Diffie-Hellman tuple. This can be checked by solving the DDH problem. Note that the key generation and signing algorithms are very simple and efficient: a signature can be computed in time $O(\log p) + T(|M|)$, where $T$ is the time needed to compute $h$. The verification algorithm is more complex, since it involves solving one DDH instance. For a $(\tau, t, \epsilon)$-GDH group, this takes $\tau$ steps.

# 4 Security definition

We will show that the gap signature scheme is secure against existential forgery under chosen-message attacks in the random oracle model. Here, the attacker can obtain signatures on messages of its choice. The goal of the attacker is to produce a forged signature—that is, a message different from all the queries and a valid signature to this message. In the random oracle model, the hash function used in the security scheme is modelled as a random function. Both the legitimate users and the attacker are assumed to have oracle access to the hash function. We once again warn the reader that the soundness of the random oracle model is problematic [4].

The security requirement is formalised as follows. We first define the advantage of an attacker, i.e. the success probability of an algorithm trying to forge a signature.

**Definition 4.** The advantage of an algorithm $F$ in existentially forging a signature scheme $S$ under a chosen-message attack in the random oracle model is

$$\mathrm{adv}_F^S = \Pr[(s, v) \leftarrow \mathrm{keyGen} : \mathrm{verify}(v, F^{S,h}(v))] \ ,$$

where the probability is taken over the internal coin tosses of the key generation algorithm and $F$. Here, $F^{S,h}(v)$ denotes the output of $F$ on input $v$ given access to a signing oracle $S$ and a hash oracle $h$. If $(M, \sigma) = F^{S,h}(v)$, it is required that $M$ is different from all the queries to $S$.

**Definition 5.** An algorithm $F$ $(t, q_h, q_s, \epsilon)$-breaks the signature scheme $S$, if $F$ runs in time at most $t$, makes at most $q_h$ queries to the hash oracle and at most $q_s$ queries to the signing oracle, and $\mathrm{adv}_F^S \geq \epsilon$.

**Definition 6.** A signature scheme is $(t, q_h, q_s, \epsilon)$-secure against existential forgery under chosen-message attacks if no algorithm $(t, q_h, q_s, t)$-breaks it.

# 5 Security proof

We are now ready to state and prove the main result in this survey.

**Theorem 7.** Let $G$ be a $(\tau, t', \epsilon')$-GDH group. Then the gap signature scheme in $G$ is $(t, q_h, q_s, \epsilon)$-secure against existential forgery under a chosen-message attack in the random oracle model, where $t \leq t' - 2c\log|G|(q_h + q_s)$, $\epsilon \geq 2eq_s\epsilon'$ and $c$ is a small constant.

The proof based on [3] follows in stages.

## 5.1 Overview

Assume that $F$ $(t, q_h, q_s, \epsilon)$-breaks the gap signature scheme. We will use $F$ to construct an algorithm $A$ that $(\tau, t', \epsilon')$-breaks the computational Diffie-Hellman problem in $G$, where $t'$ and $\epsilon'$ are as in the theorem. Since this is the counter-positive of the theorem, the result follows.

The algorithm $A$ is given a computational Diffie-Hellman instance $(g, g^a, g^b)$ and is supposed to compute $g^{ab}$. $A$ uses the CDH instance to construct a public key for the gap signature scheme. It then emulates and attack by $F$ on this key by running $F$ with oracle calls answered by $A$. When $F$ makes oracle queries for message hashes and signatures, $A$ computes an answer and passes it to $F$. When finally $F$ halts, $A$ uses the forgery output by $F$ to compute a solution to the Diffie-Hellman instance.

For convenience, we assume that $F$ is well-behaved in the following sense. We assume that $F$ always requests a hash of a message $M$ before it requests a signature for $M$, and that it always asks for the hash of the message it outputs as its forgery. This is no los of generality, since any forger $F$ can easily be modified to have this property.

The algorithm $A$ maintain a list of the oracle queries of $F$ and corresponding answers. When a message $M$ first arrives from $F$, it is assigned an index $i$. Note that $i$ is bounded by $q_h$. The message is stored in $M_i$, its hash in $m_i$, and its signature (if available) in $\sigma_i$.

## 5.2 Constructing $A$

Instead of presenting the algorithm $A$, we will describe six increasingly sophisticated variants of $A$. This will hopefully make the intuition behind $A$ clearer and makes its analysis much simpler. The last one $A_6$ will be the $A$ we are looking for. The first few variants of $A$ will be completely trivial. They will just lay down a framework for the later version and make it easy to determine the success probability. All versions except $A_6$ will only output $\top$ or $\perp$ to indicate success or failure. When $A_6$ succeeds, it will output a solution to the CDH instance. We will describe the first variant $A_1$ in detail and then list the incremental modifications needed to get the other variants.

The variants will depend on a constant $0 \leq \zeta \leq 1$ that we later will choose so that the success probability of $A$ is maximised. Let $B_\zeta$ be the probability distribution that is 1 with probability $\zeta$ and is 0 with probability $1 - \zeta$. For each message $M_i$, $A$ will pick a bit $s_i$ distributed according to $B_\zeta$. This bit will be used as a biased coin toss.

**Variant 1.** $A_1$ simply emulates a real execution of $F$. $A_1$ is given an instance $(g, g^a, g^b)$. It sets the public key to $v = g^a$. For each $1 \leq i \leq q_h$, $A_1$ pick a random bit $s_i \leftarrow B_\zeta$ and a random number $r_i \leftarrow \mathbb{Z}_p^*$, where $p = |G|$. It then sets $m_i = g^{r_i}$ and $\sigma_i = (g^a)^{r_i}$. The values $s_i$ are not used by $A_1$, but will be important for the later variants. Note that $(g, g^a, m_i, \sigma_i)$ is a valid Diffie-Hellman tuple and thus $\sigma_i$ is a valid signature for any message whose hash is $m_i$.

$A_1$ then runs $F$ on input $v$ and answers the queries of $F$ as follows. When $F$ requests a hash of a message $M_i$, $A_1$ answers with $m_i$. When $F$ request a signature on $M_i$, $A_1$ responds with $\sigma_i$. When $F$ halts, $A_1$ outputs $\top$ if $F$ succeeded in forging a signature and outputs $\perp$ otherwise.

Since the hashes $m_i$ are uniformly distributed in $G^*$, $A_1$ answers the hash queries as a random oracle. Since the signatures $\sigma_i$ are all valid, $A_1$ answers signature queries exactly as the real signature oracle would. Thus, $F$ behaves exactly as it would in a normal execution. It follows that

$$\Pr[A_1 = \top] = \mathrm{adv}_F^S \geq \epsilon \ .$$

Here, $A_1 = \top$ denotes the event that $A_1$ outputs $\top$ on input $(g, g^a, g^b)$.

**Variant 2.** $A_2$ introduces a small reduction in the success probability. This reduction is essential to be able to deviate from the perfect emulation of $F$. $A_2$ operates exactly as $A_1$, with a single exception. If $F$ fails, $A_2$ outputs $\perp$. If $F$ succeeds and outputs a forgery $(M, \sigma)$, $A_2$ outputs $\top$ if $s_i = 1$ and outputs $\perp$ otherwise, where $i$ is the index of $M$. Clearly, this modification does not affect the behaviour of $F$, so

$$\Pr[A_2 = \top] = \mathrm{adv}_F^S \cdot \Pr[s_i = 1] \geq \zeta\epsilon \ .$$

**Variant 3.** $A_3$ introduces another reduction on the success probability. The success probability of $A_3$ will be the same as that of the final version $A_6$. $A_3$ operates as $A_2$, but with one exception. If $F$ succeeds in finding a forgery $(M, \sigma)$, $A_3$ outputs $\top$ if $s_i = 1$, where $i$ is the index of $M$, and $F$ asked for signatures only on messages $M_j$ with $s_j = 0$. In all other cases, $A_3$ outputs $\perp$. Again, this modification does not affect the behaviour of $F$. If $F$ made $k$ signature queries, it is easy to see that

$$\Pr[A_3 = \top] \geq \zeta\epsilon(1-\zeta)^k \geq (1-\zeta)^{q_s}\zeta\epsilon \ .$$

**Variant 4.** $A_4$ introduces a simple shortcut. $A_4$ operates as $A_3$, except that if $F$ requests a signature for a message $M_i$ with $s_i = 1$, $A_4$ immediately halts with output $\perp$. It is easy to see that this shortcut does not affect the output distribution of $A$. Thus,

$$\Pr[A_4 = \top] \geq (1-\zeta)^{q_s}\zeta\epsilon \ .$$

**Variant 5.** $A_5$ changes the setup to better match the Diffie-Hellman instance $(g, g^a, g^b)$. $A_5$ is exactly as $A_4$, except for the setup stage. In the setup stage, $A_4$ sets $m_i = g^b g^{r_i}$ and $\sigma_i = \perp$ if $s_i = 1$, and sets $m_i = g^{r_i}$ and $\sigma_i = (g^a)^{r_i}$ (as $A_4$) if $s_i = 0$. Since $g^{r_i}$ and $g^b g^{r_i}$ are both uniformly distributed in $G^*$, and $A_4$ never actually answers a signature query for a message $M_j$ with $s_j = 1$, this does not affect the output distribution of $A$. Thus,

$$\Pr[A_5 = \top] \geq (1-\zeta)^{q_s}\zeta\epsilon \ .$$

**Variant 6.** $A_6$ finally solves the Diffie-Hellman instance $(g, g^a, g^b)$ using the information available to $A_5$. $A_6$ operates as $A_5$, but in those cases where $A_5$ outputs $\top$, $A_6$ outputs $\sigma(g^a)^{-r_i}$, where $(M, \sigma)$ is the forged signature output by $F$ and $i$ is the index of $M$. Clearly,

$$\Pr[A_6 \neq \perp] = \Pr[A_5 = \top] \geq (1-\zeta)^{q_s}\zeta\epsilon \ .$$

Moreover, $A_6$ succeeds only if $s_i = 1$, which means that $m_i = g^b g^{r_i} = g^{b+r_i}$. Since $\sigma$ is a valid signature of $M$, $(g, g^a, m_i, \sigma)$ is a valid Diffie-Hellman tuple, so $\sigma = g^{a(b+r_i)} = g^{ab}g^{r_i a}$. Thus, the output of $A_6$ is $g^{ab}$ whenever it succeeds. That is,

$$\mathrm{adv}_{A_6}^G = \Pr[A_6(g, g^a, g^b) = g^{ab}] \geq (1-\zeta)^{q_s}\zeta\epsilon \ .$$

## 5.3 Putting everything together

We showed in the previous subsection what the algorithm $A = A_6$ solves the computational Diffie-Hellman problem with probability at least $(1-\zeta)^{q_s}\zeta\epsilon$ for arbitrary $\zeta \in [0,1]$. The function $\zeta \mapsto (1-\zeta)^{q_s}\zeta\epsilon$ is maximised at $\zeta = 1/(q_s+1)$, where it has the value

$$\frac{\epsilon}{q_s+1}\left(1 - \frac{1}{q_s+1}\right)^{q_s} = \frac{\epsilon}{q_s}\left(1 - \frac{1}{q_s+1}\right)^{q_s+1} \ .$$

For large $q_s$, $\left(1 - \frac{1}{q_s+1}\right)^{q_s+1} \approx 1/e$, since $e = \lim_{x\to\infty}\left(1+\frac{1}{x}\right)^x$.

The running time of $A$ includes the running time of $F$. The additional overhead of $F$ is dominated by the oracle answers. $A$ has to compute an exponentiation for each signature and hash query. Each exponentiation can be computed using $2\log p$ group operations. Since $A$ might have to answer $q_s + q_h$ queries, its total running time is $t' \leq t + 2c\log p(q_s + q_h)$, where $c$ is a small constant accounting for the remaining overhead of $A$.

That is, if there exists a forger $F$ that $(t, q_h, q_s, \epsilon)$-breaks the gap signature scheme in $G$, there exists an algorithm $A$ that $(t', \epsilon')$-breaks the computational Diffie-Hellman problem in $G$, where $t' \leq t + 2c\log p(q_s + q_h)$ and $\epsilon' \geq \frac{\epsilon}{q_s}\left(1 - \frac{1}{q_s+1}\right)^{q_s+1}$.

Since $\left(1 - \frac{1}{q_s+1}\right)^{q_s+1} \geq \frac{1}{2e}$ for positive $q_s$, the latter inequality implies that $\epsilon \leq q_s\epsilon' \cdot 2e$. This completes the security proof. $\square$

## References


[1] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First Annual Conference on Computer and Communications Security*. ACM, 1993.

[2] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology—Crypto 2001*, volume 2139 of *LNCS*, pages 213–229. Springer-Verlag, 2001.

[3] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology—Asiacrypt 2001*, volume 2248 of *LNCS*, pages 514–532. Springer-Verlag, 2001.

[4] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. In *30th ACM Symposium on the Theory of Computation*, pages 209–218. ACM, 1998. A revisited version is available from `http://www.wisdom.weizmann.ac.il/~oded/`.

[5] Jean-Sébastien Coron. On the exact security of full domain hash. In *Advances in Cryptology—Crypto 2000*, volume 1880 of *LNCS*, pages 229–235. Springer-Verlag, 2000.

[6] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[7] Antoine Joux. A one-round protocol for tripartite Diffie-Hellman. In *Algorithmic Number Theory Symposium (ANTS-IV)*, volume 1838 of *LNCS*, pages 385–394. Springer-Verlag, 2000.

[8] Antoine Joux and Kim Nguyen. Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Cryptology ePrint Archive, Report 2001/003, 2001. `http://eprint.iacr.org/`.

[9] Alfred Menezes, Tatsuaki Okamoto, and Scott Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[10] National Institute of Standards and Technology. Digital signature standard (DSS). *FIPS*, 186–2, 2000.

[11] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Symposium on Cryptography and Information Security (SCIS 2000)*, 2000.

[12] J.H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *GMT*. Springer-Verlag, 1986.