





A Variant of BLS Signature Scheme with Tight Security Reduction

Tiong-Sik Ng¹(✉) , Syh-Yuan Tan¹ , and Ji-Jian Chin² 

¹ Faculty of Information Science and Technology,
Multimedia University, Melaka, Malaysia
ng.tiong.sik@gmail.com, sytan@mmu.edu.my

² Faculty of Engineering, Multimedia University, Cyberjaya, Malaysia
jjchin@mmu.edu.my

Abstract. In 2001, Boneh, Lynn and Shacham designed a signature scheme using the properties of bilinear pairing from elliptic curve, and based its security under the Computational Diffie-Hellman (CDH) assumption. However, the security reduction is not tight as there is a loss of roughly q_s , the number of sign queries. In this paper, we propose a variant of the BLS signature with tight security reduction based on the co-CDH assumption. Besides upgraded to the notion of strong existential unforgeability under chosen message attack, the variant is backward-compatible with the original BLS signature.

Keywords: BLS · Signature · Tight security reduction
Provable security

1 Introduction

In cryptography, a scheme is deemed secure if its security can be proven mathematically. The property of provable security was first proposed by Goldwasser and Micali in [17]. Although a cryptography primitive can be proven secure, the security reduction in the security proof may not be tight. A scheme is also said to have a tight security if breaking the scheme is as hard as solving the assumption that the scheme uses. Therefore, a tight security reduction can achieve the same security level without using larger key size. For example, the probability of breaking BLS signature is known [5] to be approximately $2e \cdot (q_s \times \varepsilon_{CDH})$ where e is the natural logarithm and q_s is the number of signatures an attacker can obtain, while ε_{CDH} is the probability of breaking the Computational Diffie-Hellman (CDH) problem. If we allow $q_s = 2^{30}$ queries, initializing BLS signature scheme with BN curve of 256 bits key size which contributes to 128 bits security, the real security of BLS is only 96 bits: $2^1 \times 2^1 \times 2^{30} \times 2^{-128} = 2^{-96}$. The non-tight security reduction shows that the BLS signature scheme has to use a larger key size, to achieve the same 128 bits security level.

The BLS is one among two well-known signature schemes that uses the shortest signature length to date, alongside the Boneh-Boyen (BB) signature

scheme [1]. The BB signature scheme is said to have a signature length as short as the BLS signature, and is also more efficient. The scheme was designed so that the bilinear pairing only needs to be done once during the verification, as compared to the BLS scheme that needs two bilinear pairings. Apart from that, the security was also proven without the help of the random oracle. However, it is known that the security of the Strong Diffie-Hellman (SDH) problem that the BB scheme uses is approximately 40% weaker compared to the CDH problem with the same parameter length, despite not using the random oracle for the security reduction [18]. Table 1 shows the comparison of ideal parameters for the BLS signature and some well-known signature schemes at 128 bits security level with their respective security tightness. Based on the table, it can be noticed that the security tightness of the RSA-PSS [8] scheme is the only one which does not contradict with its parameter size. For the signature schemes such as the BLS and BB which require bilinear pairing, we calculate the public key and signature sizes of the schemes based on the BN curve [27], where Type 3 pairing is used. Throughout this paper, point compression would be used to represent the public keys and signature lengths for schemes that are using the BN curve.

Table 1. Comparison among digital signatures at 128 bit security level

Scheme	Public key size (bits)	Signature size (bits)	Security tightness
DSA [14] [‡]	2×3072	2×256	$\varepsilon_{DSA} \approx \varepsilon_{DL}^*$
EC-DSA [14] [‡]	2×256	2×256	$\varepsilon_{EC-DSA} \approx \varepsilon_{DL}^*$
EC-DSA ⁺ [19]	2×256	2×256	$\varepsilon_{EC-DSA^+} \approx (\frac{\varepsilon_{DL}}{2q_h})^3$
Schnorr [29]	2×3072	2×256	$\varepsilon_{Schnorr} \approx (\frac{\varepsilon_{DL}}{2q_h})^3$
EC-Schnorr [20]	2×256	2×256	$\varepsilon_{EC-Schnorr} \approx 6q_h \varepsilon_{DL}$
RSA-FDH [7] [‡]	2×3072	3072	$\varepsilon_{FDH} = (q_s + q_h + 1)\varepsilon_{RSA}$
RSA-PSS [8] [‡]	2×3072	3072	$\varepsilon_{PSS} = \varepsilon_{RSA}$
BLS [22]	$2 \times 256 + 2 \times 512$	256	$\varepsilon_{BLS} = e(q_s + 1)\varepsilon_{co-CDH}$
BNN-BLS [6]	2×256	257	$\varepsilon_{BLS} = 2\varepsilon_{co-CDH}$
BB [1]	$256 + 2 \times 512 + 3072$	256	$\varepsilon_{BB} = \varepsilon_{SDH}$

* There are no proofs for these signature schemes in the random oracle model, however it is commonly believed that the probability of breaking these schemes are as hard as breaking the discrete logarithm (DL) problem.

[‡] Key size is following recommendation from NIST [26].

1.1 Related Works

After the concept of digital signatures was first proposed in [13], many digital signature schemes have emerged. Among the de-facto signature schemes are the Digital Signature Algorithm (DSA) [14] and the Schnorr signature [29]. The DSA was described based on the adoption of the ElGamal [15] signature. The DSA is a popular signature scheme that is used as a Federal Information Processing Standard, and is widely used in computer systems by non-military government

organizations. A variant of the DSA which uses the elliptic curve, also known as the EC-DSA was first proposed in [14]. The size of the EC-DSA’s public key is said to be shorter than that of the DSA, while the signature length remains the same. The Schnorr signature is another signature scheme that is based on the ElGamal signature. It was first proposed to be suitable for interactions between smart cards and terminals, as the algorithm is said to be efficient.

In [5], Boneh et al. proposed the Boneh-Lynn-Shacham (BLS) digital signature scheme based on the assumption that the Computational Diffie-Hellman (CDH) problem is intractable. As stated in their work, the signature of the BLS is proposed to be only 160 bits long compared to the RSA [28] that is 1024 bits long and the DSA that is 320 bits long, while maintaining the same security levels of the latter. Throughout the years after the first appearance of the BLS signature, many variants of the BLS have appeared. Today, the BLS signature is used for various purposes such as cloud storage [30], aggregate signatures [23], and also big data [24].

In [9], Cha and Hee Cheon designed a variant of the BLS, namely, identity-based BLS signature where the user’s public ID was used as a public key for verification. In [31], a ring signature scheme was designed by Zhang et al. which is very similar to a combination of the Cha-Cheon’s scheme and the Boneh-Boyen’s scheme. However, the scheme was not tightly secure as well. In [22], Lacharité proposed a Type-3 Pairing version of the BLS signature based on the Computational co-Diffie-Hellman Assumption (co-CDH) assumption. Her works were heavily based on Chatterjee et al.’s [10] work, where the modified co-CDH (co-CDH*) assumption was proposed.

In [16], a signature scheme very similar to the BLS was proposed by Goh and Jarecki based on the CDH assumption. Inspired by [16], Katz and Wang [21] tightened the security of FDH signatures by hashing just one bit extra together with the message. Based on the security reduction, it is shown that the scheme is almost as secure as the CDH assumption such that $\varepsilon_{FDH} = 2 \cdot \varepsilon_{CDH}$. In [6], Bellare et al. proposed an aggregate version of the BLS signature using Katz-Wang’s technique where the security reduction relies on the co-CDH assumption. The proposed BLS variant has a tight security reduction, which is similar to the result that Katz-Wang produced.

1.2 Our Contribution

In [12], Coron proposed a tight security patch for the Boneh-Franklin IBE (BF-IBE) [3] that is backward compatible based on the D-Square-BDH assumption. The upgraded BF-IBE is tightly secure with the help of a random salt drawn from the space of \mathbb{Z}_q . Inspired by Coron’s work, we propose a tight security upgrade to the BLS signature, and different from Coron’s technique, we only require the salt to be 1 bit in length. Moreover, the BLS variant is upgraded to a stronger security notion, namely, strong existential unforgeability under chosen message attacks (*seuf-cma*) based on the co-CDH assumption. Though our technique also uses a 1 bit salt, the salt is not hashed with m as in Katz-Wang’s technique [21]. Instead, it is used as an exponent for the extra public

key element. A comparison of the original BLS signature and our variant at 128 bit security level is shown in Table 2 below.

Table 2. Comparison between the original BLS and our variant

Scheme	BLS (Type-1) [5]	BLS (Type-3) [22]	BNN-BLS [6]	Our variant (Type-3)
Assumption	CDH	co-CDH	co-CDH	co-CDH
Pairing type	1	3	2	3
Public key elements	$2 \mathbb{G} $	$2 \mathbb{G}_1 + 2 \mathbb{G}_2 $	$2 \mathbb{G}_1 + 2 \mathbb{G}_2 $	$3 \mathbb{G}_1 + 2 \mathbb{G}_2 $
Signature length	$ \mathbb{G} $	$ \mathbb{G}_1 $	$ \mathbb{G}_1 + 1$ bit	$ \mathbb{G}_1 + 1$ bit
Security model	<i>euf-cma</i>	<i>euf-cma</i>	<i>seuf-cma</i>	<i>seuf-cma</i>
Security tightness	$\varepsilon_{BLS} = 2\varepsilon \cdot q_{S \in CDH}$	$\varepsilon_{BLS} = e(q_s + 1)\varepsilon_{co-CDH}$	$\varepsilon_{BNN-BLS} = 2\varepsilon_{co-CDH}$	$\varepsilon_{BLS} = \varepsilon_{co-CDH}$
Backward compatibility with original BLS	Original	Original	No	Yes

1.3 Organization

The paper is organized as such. In Sect. 2, the definitions and security model of a digital signature will be described. In Sect. 3, the original BLS signature scheme will be described in detail. Besides that, our variant will also be described alongside its security proof. In Sect. 4, the design of the variant will be discussed in detail. We conclude our findings in Sect. 5.

2 Definitions

In this section, we briefly describe the definitions and the backgrounds of digital signatures and related mathematical assumptions. Throughout this paper, we let $\{0, 1\}^*$ denote the set of all bit strings while $\{0, 1\}^n$ the set of bit strings of length n . If a string $s \in \{0, 1\}^*$ then $|s|$ denotes the length of s . If S is a set then $|S|$ denotes the size of S . Let $a \xleftarrow{R} S$ denote a randomly and uniformly chosen element a from a finite set S . Lastly let \mathbb{Z}_p denote the set of positive integers modulo a large prime p .

Definition 1. A digital signature consists of three polynomial-time algorithms: **Key Generation**, **Sign**, and **Verify**. The first two algorithms are probabilistic. The algorithms are described as follows:

1. **Key Generation** (1^k): A pair of public and secret keys are generated based on the security parameter input 1^k . The public key pk can be aired on an open channel, while the secret key sk is kept secret by the user.
2. **Sign** (m, sk): The user uses the secret key sk to sign on a message m to generate a signature, which is denoted as σ .
3. **Verify** (m, σ, pk): The verifier takes the public key pk and σ as the input to ensure that the signature is genuinely signed by the user. If the signature is authentic, the algorithm returns “True”, and “False” otherwise.

2.1 Security Notions

We refer to two security notions, the *existential unforgeability under chosen message attacks* (*euf-cma*) and *strong existential unforgeability under chosen message attacks* (*seuf-cma*). The security model of a digital signature is defined as the following:

1. **Setup.** During this stage, the Simulator \mathcal{S} generates and passes the public parameters to Adversary \mathcal{A} .
2. **Hash Query.** \mathcal{A} is allowed to make multiple hash queries on a message m in order to obtain $H(m)$.
3. **Sign Query.** \mathcal{A} is allowed to make multiple signature queries on a message m in order to obtain σ . \mathcal{S} would compute and send σ to \mathcal{A} .
4. **Forgery.** After obtaining sufficient information, \mathcal{A} would output a message and signature pair, (m^*, σ^*) , where m^* is a message that has not been signed before if it is a *euf-cma* \mathcal{A} ; else m^* is signed before but σ^* is not the previously returned signature if it is a *seuf-cma* \mathcal{A} . The forgery is successful if (m^*, σ^*) is a valid message-signature pair.

Definition 2. A digital signature scheme is $(t, q_h, q_s, \varepsilon)$ -secure against existential forgery under adaptive chosen message attacks (*euf-cma*) if for any adversary \mathcal{A} who runs in time t succeeds in forging a signature for a message that has not been signed before, i.e.

$$|\Pr[\text{Ver}(pk, m^*, \sigma^*) = 1 : (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{sk(\cdot)}}(pk); (m^*, \sigma^*) \notin \mathcal{Q}]| \leq \text{negl}(n).$$

where \mathcal{A} can make at most q_h hash queries and q_s signing queries.

Definition 3. A digital signature scheme is $(t, q_h, q_s, \varepsilon)$ -secure against strong existential forgery under adaptive chosen message attacks (*seuf-cma*) if for any adversary \mathcal{A} who runs in time t succeeds in forging a signature for a message that has previously been signed before, i.e.

$$|\Pr[\text{Ver}(pk, m, \sigma^*) = 1 : (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{sk(\cdot)}}(pk); (m, \sigma^*) \notin \mathcal{Q}]| \leq \text{negl}(n).$$

where \mathcal{A} can make at most q_h hash queries and q_s signing queries.

2.2 Bilinear Pairing

Let \mathbb{G}_1 and \mathbb{G}_2 be groups of prime order q based on the curve E over the finite field \mathbb{F}_p where $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Let g_1 be a generator of \mathbb{G}_1 and g_2 be a generator of \mathbb{G}_2 . Bilinear pairing is a function which maps elements from group \mathbb{G}_1 and group \mathbb{G}_2 to group \mathbb{G}_T , i.e. $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. The bilinear pairing function e requires the following properties:

1. Bilinearity: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
2. Non-degeneracy: $e(g_1, g_2) \neq 1$
3. e is efficiently computable, which means there is an algorithm to compute $e(g_1, g_2)$ for any $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.

2.3 Computational Assumptions

We adopt the definition of the CDH assumption from [2] as follows:

Definition 4. *Computational Diffie-Hellman (CDH) Assumption.* An algorithm \mathcal{S} is said to (t, ε) -solve the CDH problem if \mathcal{S} runs in time at most t and furthermore:

$$|\Pr[a, b \leftarrow \mathbb{Z}_q : \mathcal{S}(g, g^a, g^b) = g^{ab}]| \geq \varepsilon$$

We say that the CDH assumption is (t, ε) -hard if no algorithm (t, ε) -solves the CDH assumption.

We adopt the definition of the co-CDH assumption¹ from [10] as follows:

Definition 5. *Computational co-Diffie-Hellman (co-CDH) Assumption.* An algorithm \mathcal{S} is said to (t, ε) -solve the co-CDH problem if \mathcal{S} runs in time at most t and furthermore:

$$|\Pr[a, b \leftarrow \mathbb{Z}_q : \mathcal{S}(g_1, g_1^a, g_1^b, g_2^a) = g_1^{ab}]| \geq \varepsilon$$

We say that the co-CDH assumption is (t, ε) -hard if no algorithm (t, ε) -solves the co-CDH assumption.

Note: The relationship between the co-CDH assumption and the CDH assumption is not studied in Chatterjee et al.’s work [10]. Therefore, it is not known if there are any security gaps between the CDH assumption and the co-CDH assumption. However, it can be said that the co-CDH assumption is a Type-3 Pairing version of the CDH assumption which uses the Type-1 Pairing [6].

¹ The co-CDH assumption was first proposed by Boneh et al. in [4]. Our scheme lean towards the modified co-CDH (co-CDH*) assumption proposed by Chatterjee et al. in [10]. However, we use the co-CDH assumption throughout this paper for simplicity, as the co-CDH and co-CDH* assumptions are equivalent [10].

2.4 Pseudorandom Bit Generator

A pseudorandom bit generator is an efficiently computable function. Given an output sequence of the generator F^1 and a truly random sequence of the same length F^2 , a distinguishing algorithm \mathcal{S} cannot correctly distinguish the function with a probability of more than $1/2$, i.e.

$$\text{Adv}_{\mathcal{S}, \text{PRBG}}^{\text{prbg}}(n) = \left| \Pr_{g \xleftarrow{R} F^2} [\mathcal{S}^g = 1] - \Pr_{g \xleftarrow{R} F^1} [\mathcal{S}^g = 1] \right| = \frac{1}{2} + \varepsilon$$

where probabilities are over the choices of g and the coin tosses \mathcal{S} for non negligible ε (e.g. $\varepsilon = 1/1000$).

3 The New BLS Signature Scheme

Before presenting our upgrade to the BLS signature scheme, we first recall the original BLS signature scheme in the Type-3 pairing setting.

3.1 The BLS Signature Scheme

The BLS signature scheme [22] is defined as follows:

1. **Key Generation:** Choose generators $g_1 \xleftarrow{R} \mathbb{G}_1$ and $g_2 \xleftarrow{R} \mathbb{G}_2$, and generate a random integer $a \xleftarrow{R} \mathbb{Z}_q^*$. Then set $x_1 = g_1^a$ and $y = g_2^a$ as well as select a hash function $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$. Lastly establish the pairing function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Publish the public keys as $\{g_1, g_2, x_1, y, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, H\}$ and keep a as the secret key.
2. **Sign:** Given a message m and secret key a as input, compute the signature as $\sigma = H(m)^a$.
3. **Verify:** To verify the signature σ of a message m , check the validity of the tuple $(H(m), y, \sigma, g_2)$ by resolving $e(H(m), y) = e(\sigma, g_2)$.

For correctness, the following equation should hold:

$$\begin{aligned} e(H(m), y) &= e(H(m), g_2^a) \\ &= e(H(m)^a, g_2) \\ &= e(\sigma, g_2) \end{aligned}$$

Note: The public key x_1 is not used throughout the scheme, but it is used for the security proof in [22].

3.2 The New Construction

In order to improve the original BLS scheme, we require the addition of a bit $r \in \{0, 1\}$ and a new secret key $b \xleftarrow{R} \mathbb{Z}_q^*$. The new construction is described in detail as the following:

1. **Key Generation:** Choose generators $g_1 \xleftarrow{R} \mathbb{G}_1$ and $g_2 \xleftarrow{R} \mathbb{G}_2$, and generate random integers $a \xleftarrow{R} \mathbb{Z}_q^*$ and $b \xleftarrow{R} \mathbb{Z}_q^*$. Then set $x_1 = g_1^a$, $x_2 = g_1^b$ and $y = g_2^a$ as well as select a hash function $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$ and pseudorandom bit generator $PRBG : \{0, 1\}^* \times \mathbb{Z}_q^* \times \mathbb{Z}_q^* \rightarrow \{0, 1\}$. Lastly establish the pairing function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Publish the public keys as $\{g_1, g_2, x_1, x_2, y, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, H, PRBG\}$ and keep $\{a, b\}$ as the secret keys.
2. **Sign:** Given a message m and secret keys $\{a, b\}$ as input, generate a bit $r \leftarrow PRBG(m, a, b)$ and compute $(H(m) \cdot x_2^{-r})^a$. The signature is generated as $\sigma = (\delta, r) = ((H(m) \cdot x_2^{-r})^a, r)$.
3. **Verify:** To verify the signature $\sigma = ((H(m) \cdot x_2^{-r})^a, r)$ of a message m , check the validity of the tuple $(\delta, g_2, H(m) \cdot x_2^{-r}, y)$ by resolving $e(H(m) \cdot x_2^{-r}, y) = e(\delta, g_2)$.

For correctness, the following equation should hold:

$$\begin{aligned} e(H(m) \cdot x_2^{-r}, y) &= e(H(m) \cdot x_2^{-r}, g_2^a) \\ &= e((H(m) \cdot x_2^{-r})^a, g_2) \\ &= e(\delta, g_2) \end{aligned}$$

Note: Similar to the original BLS scheme using Type 3 pairing in [22], the public key x_1 is not used throughout the scheme. However, it is used in our security proof.

Our scheme can be used as an upgrade on the original BLS scheme, as the signing and verification algorithms are backward compatible with that of the original BLS signature algorithms. Particularly, the original signing algorithm only needs to multiply the signature $H(m)^a$ with x_2^{-ra} , while the original verification algorithm multiplies the left handside $e(H(m), y)$ with $e(x_2^{-r}, y)$. The difference with the original BLS scheme is that a “randomization” of a bit $r \leftarrow PRBG(m, a, b)$ was added² to the signature³. However, as the bit r is chosen during the generation of each signature, there is only one valid bit r for a given message⁴.

² We propose the usage of a single bit similar to Katz-Wang’s technique in [21] to optimize the signature length. However, the security proof for an integer instead of a bit r works just as well as the RSA-PFDH [11]. The security of PRBG to randomize the signature is not an issue, as proposed and used by Katz-Wang [21] and Kobitz-Menezes [19].

³ To avoid having a state where two signatures for a message exist at once where the value of the bit r may be either 0 or 1, the signer may enclose the bit r alongside σ to avoid further confusion during verification.

⁴ The value of r cannot be changed as once the signature is generated, the value of δ in the signature would be corrupted if the value of r is of a different value.

3.3 Security Proof

Theorem 1. *The new BLS signature is $(t, q_h, q_s, \varepsilon)$ -seuf-cma secure if the co-CDH assumption is (t', ε') -hard, where:*

$$\begin{aligned} \varepsilon &= \varepsilon' \\ t &= \mathcal{O}(t') \end{aligned}$$

Proof. Assume that there exists a $(t, q_h, q_s, \varepsilon)$ -adversary \mathcal{A} running in time of at most t making at most q_h hash queries and at most q_s signing queries against the new BLS scheme which forges a valid signature with probability of at least ε . We construct a simulator \mathcal{S} that solves the co-CDH problem with an advantage of at least ε' while interacting with \mathcal{A} .

Setup. \mathcal{S} receives the co-CDH challenge $\{g_1, g_1^a, g_1^b, g_2^a\}$ and must output g_1^{ab} . \mathcal{S} sets $g_1 = g_1, x_1 = g_1^a, x_2 = g_1^b$ and $y = g_2^a$. The master keys $\{a, b\}$ are not known to \mathcal{S} .

Hash Query. When \mathcal{A} submits⁵ a fresh query $H(m)$ for message m , \mathcal{S} generates random values $p_1, p_2, p \xleftarrow{R} \mathbb{Z}_q^*$, and then computes a random bit $\tilde{r} \leftarrow PRBG(m, p_1, p_2)$. \mathcal{S} then stores $\{m, p, \tilde{r}\}$ in H -list and returns $H(m) = g_1^p \cdot x_2^{\tilde{r}}$. If m was queried before, \mathcal{S} searches for the existing record from H -list and returns the same $H(m) = g_1^p \cdot x_2^{\tilde{r}}$.

Sign Query. When \mathcal{A} submits a signing query for m , we assume the hash query $H(m)$ has already been made. If not, \mathcal{S} goes ahead and computes the hash query first. In either case, \mathcal{S} can recover (p, \tilde{r}) from H -list and let $\delta = x_1^p$ to return $\sigma = (\delta, \tilde{r})$. This is a valid signature for m as:

$$\begin{aligned} \delta &= (H(m) \cdot x_2^{-\tilde{r}})^a \\ &= (g_1^p \cdot x_2^{\tilde{r}} \cdot x_2^{-\tilde{r}})^a \\ &= g_1^{ap} \\ &= x_1^p \end{aligned}$$

It should be noted that \mathcal{S} can always answer the signature queries made by \mathcal{A} .

Forgery. Without loss of generality, we assume the message m^* used in the forgery $(m^*, \sigma^* = (\delta^*, r^*))$ was queried to hash oracle. If that is not the case, \mathcal{S} issues a hash query for m^* . We distinguish the forgery of \mathcal{A} into 2 cases:

Case 1: Suppose \mathcal{A} produces a valid $(m^*, \sigma^* = (\delta^*, r^*))$ pair where the signature of m^* is never queried by \mathcal{A} before⁶, \mathcal{S} aborts if $r^* = \tilde{r}$; if $r^* \neq \tilde{r}$, \mathcal{S} goes ahead to solve the co-CDH assumption.

⁵ Different from Katz-Wang's work in [21], \mathcal{A} is not allowed to query the value of r , since it is not part of the hash inputs.

⁶ In this case, \mathcal{A} falls under the category of an *euF-cma* Adversary, whose m^* in the forgery must not be signed before.

Case 2: Suppose \mathcal{A} produces a valid $(m^*, \sigma^* = (\delta^*, r^*))$ pair where σ^* is not the response given by \mathcal{S} during the sign query⁷, \mathcal{S} goes ahead to solve the co-CDH assumption.

When $r^* \neq \tilde{r}$, \mathcal{S} can solve the co-CDH assumption by extracting g_1^{ab} as follows:

$$\begin{aligned}
\left(\frac{\delta^*}{x_1^p}\right)^{\frac{1}{(\tilde{r}-r^*)}} &= \left(\frac{(H(m^*) \cdot x_2^{-r^*})^a}{g_1^{ap}}\right)^{\frac{1}{(\tilde{r}-r^*)}} \\
&= \left(\frac{((g_1^p \cdot x_2^{\tilde{r}}) \cdot x_2^{-r^*})^a}{g_1^{ap}}\right)^{\frac{1}{(\tilde{r}-r^*)}} \\
&= \left(\frac{((g_1^p \cdot g_1^{b\tilde{r}})g_1^{-br^*})^a}{g_1^{ap}}\right)^{\frac{1}{(\tilde{r}-r^*)}} \\
&= \left(\frac{(g_1^{ap})(g_1^{ab(\tilde{r}-r^*)})}{g_1^{ap}}\right)^{\frac{1}{(\tilde{r}-r^*)}} \\
&= g_1^{ab}
\end{aligned}$$

Since \mathcal{S} can answer all hash and sign queries in either case, the probability of breaking the co-CDH assumption is:

$$\begin{aligned}
\Pr[\mathcal{S} \text{ solves co-CDH}] &= \Pr[\mathcal{A} \text{ outputs valid } \sigma^* \wedge \mathcal{S} \text{ does not abort}] \\
\varepsilon' &= \Pr[\mathcal{A} \text{ outputs valid } \sigma^*] \Pr[\mathcal{S} \text{ does not abort hash queries}] \\
&\quad \Pr[\mathcal{S} \text{ does not abort sign queries}] \Pr[r^* \neq \tilde{r}] \\
\varepsilon' &= \varepsilon \times 1 \times 1 \times 1 \\
\varepsilon' &= \varepsilon
\end{aligned}$$

Recall that \mathcal{A} is a forger in the security notion of *seuf-cma*. If m^* was previously queried to the sign oracle, the returned signature would be $\sigma = (\delta, \tilde{r})$, and so the forged signature σ^* , which is different from σ , has to be $\sigma^* = (\delta^*, r^*)$ such that $r^* \neq \tilde{r}$. Since $r \in \{0, 1\}$, $r^* \neq \tilde{r}$ happens with probability 1 and subsequently $\delta^* \neq \delta$. On the other hand, in Case 1, which is the security notion of *euf-cma*, \mathcal{A} does not query m^* to sign oracle before, and $\Pr[r^* \neq \tilde{r}] = 1/2$ happens on the forged signature σ^* . Since we are considering an upgrade to the original BLS signature scheme, we emphasize Case 2 only, which is the *seuf-cma* notion as stated in Theorem 1. The time needed to break the scheme, $t_{\mathcal{A}}$ is defined as the computation time throughout the scheme, $\mathcal{O}(t)$ and $\varepsilon' = \varepsilon$ as expected. \square

4 Discussion

4.1 Public Keys and Signature Length

To achieve a 128 bit security, the ideal size of a public key for the BLS signature on BN curve would be $2|\mathbb{G}_1| + 2|\mathbb{G}_2| = 2(|\mathbb{G}_T|/12 \times 2) + 2(|\mathbb{G}_T|/6 \times 2) =$

⁷ In this case, \mathcal{A} falls under the category of a *seuf-cma* Adversary, whose m^* in the forgery must be signed before.

$2(3072/12 \times 2) + 2(3072/6 \times 2) = 2(256 \times 2) + 2(512 \times 2) = 3072$ bits, or 1536 bits if point compression is used. However, as the original BLS scheme is not tightly secure, a larger public key of $(2(7680/12 \times 2) + 2(7680/6 \times 2))/2 = (2(640 \times 2) + 2(1280 \times 2))/2 = 7680/2 = 3840$ bits is needed to make up for the loss to achieve the same 128 bits security. A comparison of the key pairs and signature lengths between the original BLS scheme and our variant is shown in Table 3.

Table 3. Public key length and signature length

Scheme	BLS (Type-3) (ideal)	BLS (Type-3) [22]	BNN-BLS (Type-3) [6]	Our variant
Public key length	1536 bits	3840 bits	1536 bits	1792 bits
Secret key length	256 bits	384 bits	256 bits	512 bits
Signature length	256 bits	640 bits	257 bits	257 bits
Backward compatibility	Original	Original	No	Yes

Although our upgrade adds an extra \mathbb{G}_1 element to the public key, we still manage to reduce the size of the public key from 3840 bits to 1792 bits for 128 bit security due to the tight security reduction. While the original BLS signature length would be $|\mathbb{G}_1| = 640$ bits, our signature is generated in terms of $(|\mathbb{G}_1|, r)$, where the generated signature length would be $256 + 1 = 257$ bits as an additional bit is transmitted as r . Therefore, our signature length is 383 bits shorter compared to the original BLS signature at the same security level.

4.2 Tight Reduction as an Upgrade

The proposed BLS variant achieves the *seuf-cma* security besides having a tight reduction to the co-CDH problem. Although sharing the same security benefits as the Bellare et al.’s variant [6], ours is backward compatible while theirs are not. Therefore, theirs cannot be used to perform a “patching” to existing applications and standards [25] that are using the BLS signature.

For instance, our variant can be used on the aggregate BLS signatures as in [6]. Besides that, the variant can be also applied directly on Cha-Cheon’s identity based signature (IBS) scheme [9] to further tighten the security of their scheme. Moreover, the variant is also applicable for works using the BLS signature for practical applications to have a tighter security, such as for the cloud storage [30] and public auditing [31].

4.3 Coron’s BF-IBE

In [12], Coron proposed two variants of the BF-IBE based on the Decisional Square Bilinear Diffie-Hellman (D-Square-BDH) assumption and the Decisional

Bilinear Diffie-Hellman (DBDH) assumption. Both variants have a tight security with a loss of 1 bit due to the use of a random salt $y \leftarrow \mathbb{Z}_q^*$ in the user secret key. Based on our proof in Sect. 3.3, it can be noticed that our method of using a 1 bit r can be applied on Coron's method of using the random salt y as well. By doing so, the length of their user secret key can be reduced without affecting the security tightness.

5 Conclusion

In this paper, we proposed a variant of the BLS signature scheme with a tight security reduction based on the co-CDH assumption. The new scheme has backward compatibility property and can be imposed directly on the original BLS signature scheme as an upgrade. Besides that, our scheme is upgraded to a stronger security notion compared to the original BLS scheme.

Acknowledgment. The authors would like to thank the Malaysia government's Fundamental Research Grant Scheme (FRGS/2/2014/ICT04/MMU/03/1) for supporting this work.

References

1. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_4
2. Bao, F., Deng, R.H., Zhu, H.F.: Variations of Diffie-Hellman problem. In: Qing, S., Gollmann, D., Zhou, J. (eds.) ICICS 2003. LNCS, vol. 2836, pp. 301–312. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39927-8_28
3. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
4. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_26
5. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_30
6. Bellare, M., Namprempe, C., Neven, G.: Unrestricted aggregate signatures. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 411–422. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73420-8_37
7. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of 1st ACM Conference on Computer and Communications Security – ACM CCS 1993, pp. 62–73. ACM (1993)
8. Bellare, M., Rogaway, P.: The exact security of digital signatures-how to sign with RSA and Rabin. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68339-9_34

9. Choon, J.C., Hee Cheon, J.: An identity-based signature from gap Diffie-Hellman groups. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 18–30. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36288-6_2
10. Chatterjee, S., Hankerson, D., Knapp, E., Menezes, A.: Comparing two pairing-based aggregate signature schemes. *Des. Codes Cryptogr.* **55**(2), 141–167 (2010). Springer
11. Coron, J.-S.: Optimal security proofs for PSS and other signature schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_18
12. Coron, J.S.: A variant of Boneh-Franklin IBE with a tight reduction in the random oracle model. *Des. Codes Cryptogr.* **50**(1), 115–133 (2009)
13. Diffie, W., Hellman, M.: New directions in cryptography. *IEEE Trans. Inf. Theory* **22**, 644–654 (1976)
14. Kerry, C.F., Director, C.R.: FIPS PUB 186-4 Federal Information Processing Standards Publication Digital Signature Standard (DSS), FIPS Publication (2013)
15. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_2
16. Goh, E.-J., Jarecki, S.: A signature scheme as secure as the Diffie-Hellman problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 401–415. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_25
17. Goldwasser, S., Micali, S.: Probabilistic encryption. *J. Comput. Syst. Sci.* **28**(2), 270–299 (1984). Springer, Heidelberg
18. Kobitz, N., Menezes, A.: Another look at “Provable Security”. II. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 148–175. Springer, Heidelberg (2006). https://doi.org/10.1007/11941378_12
19. Kobitz, N., Menezes, A.J.: The random oracle model: a twenty-year retrospective. *Des. Codes Cryptogr.* **77**(2–3), 587–610 (2015)
20. Kiltz, E., Masny, D., Pan, J.: Optimal security proofs for signatures from identification schemes. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 33–61. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_2
21. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: ACM – CCS 2003, pp. 155–164 (2003)
22. Lacharité, M.S.: Security of BLS and BGLS signatures in a multi-user setting. In: *Advances in Cryptology 2016 – ARCTICRYPT 2016*, vol. 2, pp. 244–261. Springer, Heidelberg (2016)
23. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_28
24. Liu, C., Ranjan, R., Zhang, X., Yang, C., Georgakopoulos, D., Chen, J.: Public auditing for big data storage in cloud computing—a survey. In: 2013 IEEE 16th International Conference on Computational Science and Engineering (CSE), pp. 1128–1135 (2013)
25. Moody, D., Peralta, R., Perlner, R., Regenscheid, A., Roginsky, A., Chen, L.: Report on pairing-based cryptography. *J. Res. Nat. Inst. Stand. Technol.* **120**, 11–27 (2015)
26. Barker, E., Barker, W., Burr, W., Polk, W., Smid, M.: Recommendation for key management—part 1: general (revised.) In: NIST Special Publication (2006)

27. Pereira, G.C., Simplicio, M.A., Naehrig, M., Barreto, P.S.: A family of implementation-friendly BN elliptic curves. *J. Syst. Softw.* **84**(8), 1319–1326 (2011)
28. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21**(2), 120–126 (1978). ACM
29. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) *CRYPTO 1989*. LNCS, vol. 435, pp. 239–252. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_22
30. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling public verifiability and data dynamics for storage security in cloud computing. In: Backes, M., Ning, P. (eds.) *ESORICS 2009*. LNCS, vol. 5789, pp. 355–370. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04444-1_22
31. Zhang, F., Safavi-Naini, R., Susilo, W.: An efficient signature scheme from bilinear pairings and its applications. In: Bao, F., Deng, R., Zhou, J. (eds.) *PKC 2004*. LNCS, vol. 2947, pp. 277–290. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24632-9_20