# Fast Digital Signature Schemes as Secure as Diffie-Hellman Assumptions

Changshe Ma[1], Jian Weng[2] and Dong Zheng[2]
[1]School of Computer, South China Normal University, Guangzhou, China, 510631
changshema@gmail.com
[2]Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai, China, 200030

January 22, 2007

**Abstract:** This paper presents two fast digital signature schemes based on Diffie-Hellman assumptions. In the random oracle model, the first scheme S1 has a tight security reduction to the computational Diffie-Hellman (CDH) problem; and the second scheme S2 has a tight security reduction to the decisional Diffie-Hellman (DDH) problem.

Comparing with existing signature schemes (whose security is tightly related to CDH problem) like *EDL* signature schemes, the signature generation of S1 is about 27% faster, and the verification is about 35% faster, if without considering the hash function evaluations. Comparing with existing signature schemes (whose security is tightly related to DDH problem) like KW-DDH signature scheme, the signing of S2 is about 40% faster and the verification is about 35% faster.

The high efficiency of the proposed schemes is attributed to a new protocol EDL_MWZ which implements the proof of equality of discrete logarithm. The EDL_MWZ protocol outperforms its counterpart, the Chaum and Pedersen protocol, as its computation is about 38% faster and its bandwidth is $|G|$ bits shorter. This new protocol may be of independent interests.

**Key Words:** Public-key cryptography, signature schemes, discrete logarithm problem, Diffie-Hellman problem, tight reduction.

## 1 Introduction

Digital signature has been the most charming research field of public key cryptography since it was firstly introduced by Diffie and Hellman [13] in 1976. According to the IEEE P1363 [28] standards, One of the standard mathematical settings to construct cryptographic algorithms is the discrete logarithm (DL) [32] setting, based on which, a vast variety of signature schemes [2, 26, 18, 19, 14, 31] have been presented in literature.

Nowadays, provable security is a widely accepted necessary requirement for cryptographic primitives. The core technique of provable security is the security reduction. Typically, there are two types of reduction: tight reduction and loose reduction [24]. The tight reduction says that we can transfer an adversary breaking the cryptographic primitive into a solver to solve some difficult problem in a probability close to 1 [2].

In the random oracle model [5, 10], the first efficient signature scheme tightly related to RSA [30] assumption is [6]. Up to now, the known shortest signature scheme is BLS scheme due to Boneh, Lynn and Shacham [7], but it is limited to the groups with pairing. In the standard model, RSA-based signature schemes [12, 17] have been shown to be secure under stronger assumptions. Based on pairing, Boneh and Boyen [1] proposed an efficient short signature scheme which is secure tightly related to $q$-strong Diffie-Hellman assumption.

The traditional discrete logarithm based signature schemes, such as El Gamal [14], DSS [26], and Schnorr [31] signature schemes, either suffer a loose security reduction or require non-standard assumptions. Recently, Goh and Jarecki [19] showed that a previously introduced signature scheme called *EDL* signature scheme [9] has a tight security reduction related to CDH assumption. The disadvantage of *EDL* signature scheme is that its signature size is bigger than its traditional counterparts and needs more exponentiations. The signature size of *EDL* scheme has been shortened by Katz and Wang [22] (we call it as KW-CDH). In the same paper, they also proposed a more efficient signature scheme (we call it as KW-DDH) with tight security reduction to decisional Diffie-Hellman (DDH) problem [23] which is a stronger assumption than the CDH problem. In [2], a new signature scheme (we call it as CM signature scheme) was proposed. Similar to *EDL* signature scheme, CM signature scheme also has a tight security reduction to CDH problem, but it has a smaller signature size, especially it provides a fast online computation [33].

Discrete exponentiation is the most expensive computational component in the public key cryptography. It dominates the computational cost of public key cryptographic schemes. Much work goes into improving the performance of cryptographic algorithms by reducing the number of exponentiations. Especially, in a mobile environment, exponentiation would consume the power the battery which is very limited. Reducing the number of exponentiation will increase battery life. So, the fewer the number of exponentiations, the better. This is a domain where a 10% improvement would be very welcome and a 50% improvement would be dramatic.

On the other hand, researchers engage in studying optimal and fast exponentiation algorithms[3], which includes fast single exponentiation algorithms and fast multi-exponentiation algorithms [4]. Interestingly, the cost of multi-exponentiation is not equal to that of a single exponentiation multiplying the number of powers [4]. Indeed, the cost of a multi-exponentiation is only a little more than that of a single exponentiation. For example, the cost of a two-element multi-exponentiation is about 20% more than that of a single exponentiation [19]. This gap will help us construct fast signature schemes by merging single exponentiations into multi-exponentiation.

| **Scheme** | **Sign** | **Verify** | **Signature Size** | $\|\mathbf{pk}\|$ | **Assumption** |
|---|---|---|---|---|---|
| *EDL*[**19**] | 3 exp | 2 mexp | $\|\mathbb{G}\| + 2\|\mathbb{Z}_q\| + \|r\|$ | $1\|\mathbb{G}\|$ | CDH |
| KW-CDH[22] | 3 exp | 2 mexp | $\|\mathbb{G}\| + 2\|\mathbb{Z}_q\| + 1$ | $1\|\mathbb{G}\|$ | CDH |
| KW-DDH[22] | 2 exp | 2 mexp | $2\|\mathbb{Z}_q\|$ | $4\|\mathbb{G}\|$ | DDH |
| CM[2] | 3 exp | 2 mexp | $\|\mathbb{G}\| + 2\|\mathbb{Z}_q\|$ | $1\|\mathbb{G}\|$ | CDH |
| our scheme S1 | 1 exp + 1 mexp | 1 mexp | $\|\mathbb{G}\| + 2\|\mathbb{Z}_q\| + 1$ | $1\|\mathbb{G}\|$ | CDH |
| our scheme S2 | 1 mexp | 1 mexp | $2\|\mathbb{Z}_q\|$ | $4\|\mathbb{G}\|$ | DDH |

**Table 1:** Comparisons amongst signature schemes with tight security reduction to Diffie-Hellman assumptions. Where, "exp" stands for an exponentiation in the group, and "mexp" for a multi-exponentiation. We make the comparisons according to the signing cost, verification cost, signature size, public key size and the security related assumptions. Note that we do not consider the cost of hash function evaluations. Averagely, hashing to $\mathbb{G}_{g,q}$ is less efficient than a single exponentiation.

## 1.1 Our Contributions

In this paper, we first propose a fast protocol EDL_MWZ to implement the proof of equality of logarithm. Its outstanding advantage over Chaum and Pedersen's protocol [11] is the high efficiency. Since the EDL_MWZ protocol needs only two multi-exponentiations, while Chaum and Pedersen's protocol needs two single exponentiations and two multi-exponentiations. These implies that there is a 38% computational improvement of our protocol. Furthermore, the bandwidth of EDL_MWZ protocol is $|\mathbb{G}|$ bits shorter. The protocol EDL_MWZ may be of independent interests and can improve a variety of cryptographic schemes like threshold signature/decryption schemes, verifiable secret sharing schemes, etc..

Based on the EDL_MWZ protocol, we proposed two fast digital signature schemes S1 and S2 whose security is also tightly related to Diffie-Hellman assumptions in the random oracle model. Security of our scheme S1 relies on the hardness of CDH problem; security of our scheme S2 (which is more efficient than S1) relies on the hardness of DDH problem. Thanks to the advantage of EDL_MWZ protocol, the derived signature schemes S1 and S2 need fewer exponentiations. The efficiency comparisons between our schemes and other existing schemes are detailed as follows (the advantages of our schemes can be easily seen from the above table 1).

Comparing with existing signature schemes (whose security is tightly related to CDH problem) like *EDL* signature schemes, the signature generation of S1 is about 27% faster; and the verification is about 35% faster, if without considering the hash function evaluations. Concretely, only one exponentiation and one multi-exponentiation is required for signature generation; and one multi-exponentiation is required for verification of our scheme S1. Hence, there is a two-exponentiation reduction in our scheme S1.

Comparing with existing signature schemes (whose security is tightly related to DDH problem) like KW-DDH signature scheme, the signing of S2 is about 40% faster and the verification is about 35% faster. Concretely, only one multi-exponentiation is required separately for signing and verification of our scheme S2. So, there is also a two-exponentiation reduction in our scheme S2.

### 1.2 Outline of the Paper.

The rest of the paper is organized as follows. We first recall some definitions and notations in § 2. In § 3, we propose the new protocol EDL_MWZ to implement the proof of equality of discrete logarithm. In § 4, we present and analyze the new signature scheme S1. In § 5, we present and analyze the new signature scheme S2. In § 6, we draw a conclusion.

## 2 Preliminaries

In this section, we will present the definition of the signature scheme and its security model. We also introduce some mathematical notations and the difficult problems. At first, we recall the definition of the signature scheme and its security model.

### 2.1 Signature Scheme

Generally, a signature scheme S=(KeyGen, Sign, Verify) consists of a triple of algorithms which are described as follows.

**KeyGen** Key generation algorithm: on input $1^k$, it outputs a key pair $(sk, pk)$ of matching the private key and the corresponding public key.

**Sign** Signature generation algorithm: on input a message $m$ and the user's private key $sk$, it outputs a signature $\sigma$ on message $m$.

**Verify** Signature verification algorithm: on input a signature $\sigma$, a message $m$ and a public key $pk$, it outputs 1 to imply a valid signature or a symbol $\perp$ to imply an invalid signature.

A universally accepted security notion of a signature scheme is existential unforgability under adaptive chosen message attack (EUF-CMA) [21] in the random oracle model. Under such a model, a forger can ask the sign oracle in an adaptive fashion. Its goal is to produce a valid signature on a message that has not been asked to the sign oracle. This model is described as the following game EUF-CMA played between a challenger and a forger $\mathcal{F}$. Our computing model is the probabilistic polynomial time Turing machine.

**Game EUF-CMA.**

***Initialization*** The challenger runs the key generation algorithm KeyGen to generate a public/private key pair $(pk, sk)$, $sk$ is kept secret while $pk$ is given to the forger $\mathcal{F}$.

***Queries*** $\mathcal{F}$ performs a series of oracle queries in an adaptive fashion. The following queries are allowed:

***Sign oracle queries*** in which $\mathcal{F}$ submits a message $m \in \mathcal{M}$ to the challenger and obtains a signature $\sigma$ on message $m$ under the public key $pk$.

***Hash queries*** in which $\mathcal{F}$ submits a string and obtains its corresponding hash value (here, we deal with the hash function as ideally random function).

***Output*** At the end of the game, $\mathcal{F}$ outputs a message and signature pair $(m^*, \sigma^*)$. We say that $\mathcal{F}$ wins the game if $(m^*, \sigma^*)$ is a valid message-signature pair with the restriction that $m^*$ has never been asked to the sign oracle.

The above game describes a security model for signature unforgeability. $\mathcal{F}$'s advantage is defined to be $\mathsf{Adv}(\mathcal{F}) = \Pr[\mathcal{F}$ wins the game EUF-CMA$]$.

**Definition 2.1.** We say that an algorithm $(q_H, q_S, t, \epsilon)$-breaks the signature scheme if in the above game there exists a forger running in a time at most $t$ and making at most $q_H$ hash functions queries, $q_S$ sign oracle queries with $\mathsf{Adv}(\mathcal{F}) \geq \epsilon$.

A signature scheme is said to be $(q_H, q_S, t, \epsilon)$-secure if there exists no such a forger which can $(q_H, q_S, t, \epsilon)$-break it.

## 2.2 Mathematical Preliminaries

We consider the mathematical preliminaries for constructing and proving our signature schemes. Let $\mathbb{G}$ be an abelian group. Also let $\mathbb{G}_{g,q} \subset \mathbb{G}$ denote a cyclic subgroup generated by $g$, whose order is a large prime $q$.

Let $x$ be a random number in $\mathbb{Z}_q$. Define $y = g^x$. The discrete logarithm problem says that given $y$ and $g$ to find $x \in \mathbb{Z}_q$ such that $g^x = y$. In this paper, the discrete logarithm of $y$ w.r.t. $g$ will be denoted as $\mathrm{Dlog}_g y = x$. We assume that the discrete logarithm problem over $\mathbb{G}$ is hard.

**Computational Diffie-Hellman (CDH) Problem.** For random numbers $a$ and $b$, given $g^a, g^b \in \mathbb{G}$, to compute $g^{ab} \in \mathbb{G}$. An algorithm $\mathcal{A}$ has advantage $\epsilon$ in solving CDH problem in group $\mathbb{G}$ if

$$\Pr[\mathcal{A}(g, g^a, g^b) = g^{ab}] \geq \epsilon$$

where the probability is over the random choice of generator $g \in \mathbb{G}$, the random choice of $a$ and $b$, and the coin toss of $\mathcal{A}$.

**Definition 2.2.** We say that $(t, \epsilon)$-CDH assumption holds in $\mathbb{G}$ if no polynomial time algorithm runs in time at most $t$, and has advantage at least $\epsilon$ in solving CDH problem in $\mathbb{G}$.

**Decisional Diffie-Hellman (DDH) Problem.** For random numbers $a, b, c \in \mathbb{Z}_q^*$, informally, the decisional Diffie-Hellman problem is to distinguish between the tuple of the form $(g, g^a, g^b, g^{ab})$ and the tuple of the form $(g, g^a, g^b, g^c)$. A tuple $(g, g^a, g^b, g^c)$ is called a "DDH tuple" if and only if it satisfies that $ab = c$, otherwise, it is called a "random tuple". Formally, A distinguisher $\mathcal{D}$ is said to have advantage $\epsilon$ in solving the DDH problem in $\mathbb{G}$ if

$$|\Pr[\mathcal{D}(g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{D}(g, g^a, g^b, g^c) = 1]| \geq \epsilon,$$

where the probability is over the random choice of generator $g \in \mathbb{G}$, the random choice of $a, b$ and $c$, and the coin toss of $\mathcal{D}$.

**Definition 2.3.** We say that $(t, \epsilon)$-DDH assumption holds in group $\mathbb{G}$ if no polynomial time distinguisher runs in time at most $t$, and has advantage at least $\epsilon$ in solving the DDH problem in $\mathbb{G}$.

## 3 Proof of the Equality of Discrete Logarithm

In this section, we first review Chaum and Pedersen's protocol [11]. Then, we will introduce a new zero knowledge proof protocol to prove the equality of discrete logarithm.

Let $g, h \in \mathbb{G}_{g,q}$ be two publicly known generators. The prover selects a secret $x \in \mathbb{Z}_q$ and computes the public values $y_1 = g^x$ and $y_2 = h^x$. The prover must convince the verifier that:

$$\mathrm{Dlog}_g\ y_1 = \mathrm{Dlog}_h\ y_2.$$

### 3.1 Chaum and Pedersen's Protocol

The protocol, described by Chaum and Pedersen [11], is as follows:

1. The prover randomly chooses $t \in \mathbb{Z}_q^*$ and sends the pair $(a, b) = (g^t, h^t)$ to the verifier.
2. After receiving the pair, the verifier randomly selects a challenge $c \in \mathbb{Z}_q^*$ and sends it to the prover.
3. The prover computes $s = t - cx \bmod q$ and sends $s$ to the verifier.
4. The verifier accepts the proof if:

$$a = g^s y_1^c \text{ and } b = h^s y_2^c.$$

It is well known that the above protocol is honest verifier zero knowledge. This protocol has been used to design a variety of schemes like the signature schemes with tight security reduction, verifiable encrypted signature, threshold cryptography, etc.. Its performance improvement will make all of those schemes better, as it is the most expensive component of computational cost.

## 3.2 The New Protocol

Here, we present a new protocol EDL_MWZ to prove the equality of discrete logarithm. The trick lies behind Chaum and Pedersen's protocol is that the elements $a$ and $b$ have the same form of multi-exponentiation. In our protocol, the elements $a$ and $b$ are merged into a whole unity, which will result in a two-exponentiation reduction. We use a new technique to prevent an attack which is very similar to rogue-key attack in the multi-signature schemes [8, 27]. So, the technique presented here can be used to design multi-signature schemes in the plain public key model [8]. The new protocol proceeds as follows:

In the EDL_MWZ protocol, we assume that the discrete value of $\mathrm{Dlog}_g h$ is unknown to all participants. Let $H : \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$ be a random oracle.

1. The prover selects uniformly at random $t \in \mathbb{Z}_q^*$ and sends the commitment $v = (g^n h)^t$ to the verifier, where $n = H(g, h, y_1, y_2)$.
2. After receiving $v$, the verifier selects a random challenge $c \in \mathbb{Z}_q^*$ and sends it to the prover.
3. The prover sends the response $s = t - cx \bmod q$ to the verifier.
4. The verifier accepts the proof if:

$$v = (g^n h)^s (y_1^n y_2)^c, \text{ where } n = H(g, h, y_1, y_2).$$

The above described protocol requires only two multi-exponentiations comparing with two exponentiations and two multi-exponentiations for Chaum and Pedersen's protocol. The cost of a four-element multi-exponentiation is about 55% more than that of a single exponentiation. Hence, our protocol offers a saving of 38% computation cost. In addition, the bandwidth of EDL_MWZ protocol is $|G|$ bits less than that of Chaum and Pedersen's protocol. The proposed protocol is an honest verifier zero-knowledge proof:

**Lemma 1.** *In the random oracle model, if the discrete logarithm value of* $\mathrm{Dlog}_g h$ *is unknown to all participants, then the protocol* EDL_MWZ *is an honest verifier statistically zero knowledge proof.*

**Proof.** The detailed proof has been given in the Appendix. Here, we only state a sketch of the proof. The completeness can be verified without any hardness. The soundness can be achieved by using the general forking lemma [8] and the rewinding technique. The simple trick is that if an adversary $\mathcal{A}$ can cheat the verifier with non-negligible probability, then we can translate it into an algorithm $\mathcal{B}$ to extract the discrete logarithm value $\mathrm{Dlog}_g h$. The protocol is zero knowledge because the view of the verifier can be easily simulated by selecting randomly $e, s \in \mathbb{Z}_q^*$ and computing $v = (g^n h)^s (y_1^n y_2)^e$.

The above presented interactive protocol can be transformed into a non-interactive protocol through Fiat-Shamir heuristic [16]. The verifier's challenge $e$ is replaced with the hash value of the commitment $v$ and of the public data using a collision resistance hash function $G$. So,

*a non-interactive protocol of proof of equality of discrete logarithm is a pair* $(e, s)$*, where* $e = G(g, h, y_1, y_2, v)$.

Indeed, this non-interactive protocol is also an honest verifier statistically zero-knowledge proof:

**Lemma 2.** *In the random oracle model, if the discrete logarithm value of* $\mathrm{Dlog}_g h$ *is unknown to all participants, then the above protocol is honest verifier statistically zero knowledge.*

**Proof.** It can be done very similar to Lemma 1.

## 4 A Signature Scheme Based on the CDH Assumption

In this section, we present a new signature scheme S1=(KeyGen, Sign, Verify) from abelian groups. To describe our scheme, some global parameters are required to be defined in advance. Let $\ell_p$ and $\ell_q$ denote the security parameters. Let also $\mathbb{G}_{g,q}$ be a subgroup of $\mathbb{G}$, where $|q| = \ell_q$ and representation of the elements in $\mathbb{G}_{g,q}$ is included in $\{0,1\}^{\ell_p}$. Let $H : \{0,1\}^* \longrightarrow \mathbb{G}_{g,q}$, $L : \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$ and $G : \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$ be three hash functions. The new signature scheme S1 is defined as follows.

**KeyGen** Every user picks his random private key $x$ from $\mathbb{Z}_q^*$ and computes his public key $y = g^x$.

**Sign** The inputs are a secret key $x \in \mathbb{Z}_q$ and a message $m \in \{0,1\}^*$. At first, the signer checks whether $m$ has been signed before. If so, output the previously generated signature. Otherwise, it does as below.

1. pick a random number $k$ from $\mathbb{Z}_q^*$ and a random bit $b_m \in \{0,1\}$;

2. compute $h = H(m, b_m) \in \mathbb{G}_{g,q}$;
3. compute $u = h^x$;
4. compute $v = (g^n h)^k$, where $n = L(m, g, h, y, u)$;
5. compute $r = G(m, g, h, y, u, v)$;
6. compute $s = k - xr \bmod q$.

The signature on message $m$ is $\sigma = (u, r, s, b_m)$.

**Verify** To verify a signature $\sigma$ on a message $m$, one does as follows.

1. parse $\sigma$ as $(u, r, s, b_m)$;
2. compute $h = H(m, b_m)$ and $n = L(m, g, h, y, u)$;
3. compute $v' = (g^n h)^s (y^n u)^r$;
4. verify whether $r = G(m, g, h, y, u, v')$?

If the equation in step 4 holds, then return 1, else return $\perp$.

The consistency of our scheme can be easily verified as

$$v' = (g^n h)^s (y^n u)^r = (g^n h)^s (g^n h)^{xr} = (g^n h)^{s+xr} = (g^n h)^k = v.$$

It is the case that $r = G(m, g, h, y, u, v')$. We now prove the security.

**Theorem 1** *In the random oracle model, if there exists an adversary $\mathcal{F}$ that has a non-negligible advantage $\epsilon$ against the unforgeability of the scheme S1 when running in a time $t$, making at most $q_S$ Sign queries, $q_H$ queries on hash function $H$, $q_L$ queries on hash function $L$ and $q_G$ queries on oracle $G$, then there exists an algorithm $\mathcal{B}$ that can solve the CDH problem with a probability*

$$\epsilon' \geq \frac{1}{2}\epsilon - \eta - \frac{3}{q} \text{ in a time } t' < t + 3.55(q_S + q_H + q_L + q_G)t_e,$$

*where $\eta$ denotes the probability to break the soundness of the protocol* EDL_MWZ *and $t_e$ denotes the time required for an exponentiation evaluation.*

**Proof.** We describe how to construct an algorithm $\mathcal{B}$ that runs $\mathcal{F}$ as a subroutine to solve the CDH problem. Let $\mathbb{G}_{g,q}$ be a cyclic group and $(g, g^a, g^b)$ a CDH problem instance. $\mathcal{B}$ runs $\mathcal{F}$ with the public key $y = g^a$ and the public parameters $(\mathbb{G}_{g,q}, g, q)$. Then, $\mathcal{F}$ performs adaptive queries which are handled by $\mathcal{B}$ as follows.

**Hash queries.** To simulate hash queries, $\mathcal{B}$ maintains three hash query lists $\mathcal{L}_H$, $\mathcal{L}_L$, $\mathcal{L}_G$ and a signature list $\mathcal{L}_S$. All these lists are initially empty. Assuming that all hash queries do not repeat. When $\mathcal{F}$ asks for a hash query (may be on $H, L$ or $G$) on a point where a message $m$ is contained, $\mathcal{B}$ first checks whether $m$ has been contained in the list $\mathcal{L}_S$. If has not, then $\mathcal{B}$ does as the following *signature generation procedure*:

1. select four random numbers $(k, n, r, s)$ from $\mathbb{Z}_q^*$ and a random bit $b_m$ from $\{0, 1\}$;
2. compute $u = y^k$;
3. set $h = H(m, b_m) = g^k$;
4. compute $v = (g^n h)^s (y^n u)^r$;
5. set $L(m, g, h, y, u) = n$ and $G(m, g, h, y, u, v) = r$ and the signature on message $m$ be $\sigma = (u, r, s, b_m)$;
6. update lists $\mathcal{L}_H, \mathcal{L}_L, \mathcal{L}_G$ and $\mathcal{L}_S$ respectively.

Next, $\mathcal{B}$ also does as follows (note that in this case, $\mathcal{L}_S$ definitely contains the message $m$ and its signature):

1. If the query is $H(m, b)$, $\mathcal{B}$ checks whether $b = b_m$. If so, return $h^k$. Otherwise, $\mathcal{B}$ selects a random number $z \in \mathbb{Z}_q$ and returns $(g^b)^z$ as the answer and inserts the tuple $(m, b, (g^b)^z, z)$ into the list $\mathcal{L}_H$.
2. If the query is on hash function $L$ or $G$, $\mathcal{B}$ manipulates it in a natural way: $\mathcal{B}$ first checks whether the querying point has been contained in lists $\mathcal{L}_L$ or $\mathcal{L}_G$. If so, the previously assigned value is returned. Otherwise, $\mathcal{B}$ returns an element chosen uniformly at random from $\mathbb{Z}_q^*$ and updates the corresponding list.

**Sign queries.** When a signature query on message $m$ is asked, $\mathcal{B}$ first determines whether $m$ has been contained in the list $\mathcal{L}_S$. If so, return the previously generated signature. Otherwise, $\mathcal{B}$ follows the above *signature generation procedure* and returns the resulted signature to $\mathcal{F}$.

As we can see, this simulator is valid and indistinguishable from an actual signer. Thus, the probability that $\mathcal{F}$ outputs a valid forgery in the simulated experiment is exactly $\epsilon$.

**Solving the CDH Problem** When $\mathcal{F}$ returning a valid signature forge $(\tilde{u}, \tilde{r}, \tilde{s}, \tilde{b})$ on message $\tilde{m}$ (we assume that $\mathcal{F}$ has asked the hash queries $H(\tilde{m}, *)$, $L(\tilde{m}, *)$ and $G(m, *)$ before), $\mathcal{B}$ first checks whether $\tilde{b} = b_{\tilde{m}}$. If so, $\mathcal{B}$ aborts with output "failure" (Since $\mathcal{F}$ did not previously request the signature on message $\tilde{m}$, the value of $b_{\tilde{m}}$ is independent of the view of $\mathcal{F}$. So, the probability that $\tilde{b} = b_{\tilde{m}}$ is $\frac{1}{2}$). Otherwise, it searches the list $\mathcal{L}_H$ to find the tuple $(\tilde{m}, \tilde{b}, \tilde{z}, (g^b)^{\tilde{z}})$ which indicates that $\tilde{h} = H(\tilde{m}, \tilde{b}) = (g^b)^{\tilde{z}}$. Then, $\mathcal{B}$ forms the following equation

$$\tilde{u} = (H(\tilde{m}, \tilde{b}))^a = ((g^b)^{\tilde{z}})^a = (g^{ab})^{\tilde{z}}$$

and extracts the solution of the given CDH instance as $\tilde{u}^{\frac{1}{\tilde{z}}}$, provided that $\mathrm{Dlog}_g y = \mathrm{Dlog}_{\tilde{h}} \tilde{u}$.

Let $\mathsf{bad}$ be the event that $\mathcal{F}$ outputs a valid signature forge $(\tilde{u}, \tilde{r}, \tilde{s}, \tilde{b})$ with $\mathrm{Dlog}_g y \neq \mathrm{Dlog}_{\tilde{h}} \tilde{u}$, which implies that $\mathcal{F}$ has successfully attacked the soundness of protocol EDL_MWZ (note that the discrete logarithm value of $\mathrm{Dlog}_g \tilde{h}$ is indeed unknown to $\mathcal{F}$). According to the lemma 2, $\Pr[\mathsf{bad}] \leq \eta$. Here, we admit that the forger has the ability to compute the discrete logarithm value $\mathrm{Dlog}_g \tilde{h}$, but its success probability can bounded by $\eta$.

Now, we give a upper bound for the probability that $\mathcal{F}$ forges a valid signature without asking hash function queries on $L$, $H$ or $G$. Since all hash functions are modeled as random oracles, it is no hard to see that $\mathcal{F}$'s success probability is no more than $\frac{3}{q}$.

Put all together, $\mathcal{B}$'s success probability $\epsilon'$ satisfies

$$\epsilon' \geq \frac{1}{2}\epsilon - \eta - \frac{3}{q}$$

and the running time $t'$ satisfies $t' < t + 3.55(q_S + q_H + q_L + q_G)t_e$, since every query relevant to a message may need two single exponentiations and one four-element multi-exponentiation to produce the signature. □

**Efficiency Consideration.** In our signature scheme, the computational costs are one exponentiations and one multi-exponentiation for the signer and one multi-exponentiations for the verifier, whereas in *EDL* and CM signature schemes, three exponentiations and two multi-exponentiations are required for signature generation and verification separately. To avoid having the signer maintain a record of all previously signed signature/message pairs, the technique of [20] can also be used for our scheme S1.

## 5 A Signature Scheme Based on the DDH Assumption

In this section, we introduce another efficient signature scheme S2. This new scheme has been proved to be secure tightly related to the DDH assmuption, rather than the CDH assumption. But it is more efficient than scheme S1, as scheme S2 has a two-exponentiation reduction comparing with S1.

As before, Let $\ell_p$ and $\ell_q$ denote the security parameters. Let also $\mathbb{G}_{g,q}$ be a subgroup of $\mathbb{G}$, where $|q| = \ell_q$ and representation of the elements in $\mathbb{G}_{g,q}$ is included in $\{0,1\}^{\ell_p}$. Let $h$ be another generator of $\mathbb{G}_{g,q}$. Also let $L : \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$ and $H : \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$ be two hash function that will be modeled as random oracles. Our second digital signature scheme S2 is defined as follows.

**KeyGen** Every user selects his random private key $x$ from $\mathbb{Z}_q^*$, and computes his public key as $(y_1, y_2) = (g^x, h^x)$.

**Sign** The inputs are a secret key $x \in \mathbb{Z}_q$ and its corresponding public key $(y_1, y_2)$, and a message $m \in \{0,1\}^*$. To generate the signature, one does as below.

1. pick a random number $k$ from $\mathbb{Z}_q^*$;
2. compute $n = H(m, g, h, y_1, y_2)$;
3. compute $v = (g^n h)^k$;
5. compute $e = G(m, g, h, y_1, y_2, v)$;
6. compute $s = k - xe \bmod q$.

The signature on message $m$ is $\sigma = (e, s)$.

**Verify** To verify a signature $\sigma$ on a message $m$, one does as follows.

1. parse $\sigma$ as $(e, s)$;
2. compute $n = H(m, g, h, y_1, y_2)$;
3. compute $v' = (g^n h)^s (y_1^n y_2)^e$;
4. verify whether $e = G(m, g, h, y_1, y_2, v')$?

If the equation in step 4 holds, then return 1, else return $\perp$.

It is not hard to see that the signature scheme S2 is correct. Since $y_1 = g^x$ and $y_2 = h^x$, the signature verification algorithm computes $v' = (g^n h)^s (y_1^n y_2)^e = (g^n)^{s+xe} h^{s+xe} = (g^n h)^k = v$. It is indeed the case that $G(m, g, h, y_1, y_2, v') = e$.

**Theorem 2** *If $(t', \epsilon')$-DDH assumption holds on the group $\mathbb{G}_{g,q}$, then the above signature scheme is $(q_H, q_s, t, \epsilon)$-secure in the random oracle model, where $\epsilon \geq \epsilon' + \eta + \frac{1}{q}$ and $t \leq t' + 1.55(q_S + 1)t_e$. (where $\eta$ denotes the probability to break the soundness of the protocol* EDL_MWZ *and $t_e$ presents the cost of a single exponentiation in group $\mathbb{G}$.)*

**Proof.** Assuming that we have a forger $\mathcal{F}$, running in time at most $t$ and making at most $q_H$ hash function queries (which includes queries to $H$ and $G$) and at most $q_S$ signature generation queries, which produces a forged signature with advantage at least $\epsilon$. We use $\mathcal{F}$ to construct an algorithm $\mathcal{D}$ to solve the DDH problem with probability at least $\epsilon'$ and running time at most $t'$. The detailed description of the algorithm $\mathcal{D}$ is as follows.

Algorithm $\mathcal{D}$ is given as input a challenge tuple $(g, h, y_1, y_2) \in \mathbb{G}_{g,q}^4$. Its goal is to determine whether this presents a "random tuple" or a "DDH tuple". To this end, it sets the public parameter as $g, h$ and $\mathbb{G}_{g,q}$ and runs $\mathcal{F}$ on the public key $(y_1, y_2)$. Algorithm $\mathcal{D}$ simulates the random oracle queries and signing queries as follows.

**Hash queries.** In order to response the queries on hash functions $H$ and $G$, $\mathcal{D}$ maintains three lists $\mathcal{L}_H$, $\mathcal{L}_G$ and $\mathcal{L}_S$ which are initially empty. The list $\mathcal{L}_S$ contains the simulated signatures generated by $\mathcal{D}$. When a query $H(m, g, h, y_1, y_2)$ (or $G(m, g, h, y_1, y_2, v)$) is asked, $\mathcal{D}$ first determines whether it has been asked before by searching the list $\mathcal{L}_H$ (or $\mathcal{L}_G$). If so, $\mathcal{D}$ returns the previously defined value. Otherwise, $\mathcal{D}$ proceeds as the following *signature generation procedure.*

1. choose three numbers $n, e, s$ uniformly at random from $\mathbb{Z}_q^*$;
2. compute $v' = (g^n h)^s (y_1^n y_2)^e$;
3. set $H(m, g, h, y_1, y_2) = n$, $G(m, g, h, y_1, y_2, v') = e$ and the signature on message $m$ be $\sigma = (e, s)$.

At last, $\mathcal{D}$ responses $\mathcal{F}$ with $n$ (or $e'$, where $e' = e$ if $v = v'$, otherwise $e'$ is a random number chosen from $\mathbb{Z}_q^*$) and updates the lists $\mathcal{L}_H$, $\mathcal{L}_G$ and $\mathcal{L}_S$ correspondingly.

**Signing queries.** When $\mathcal{F}$ asks a signature query on message $m$, $\mathcal{D}$ first determines whether it was contained in the list $\mathcal{L}_S$. If so, the previously generated signature is returned. Otherwise, $\mathcal{D}$ follows the above *signature generation procedure* and returns the resulted signature to $\mathcal{F}$.

At some point, $\mathcal{F}$ outputs a signature forger $(\sigma^*, s^*)$ on a message $m^*$ which has never been submitted for a signature query. $\mathcal{D}$ checks whether $(\sigma^*, s^*)$ is a valid signature on message $m^*$. If so, $\mathcal{D}$ outputs 1 to indicated that the challenge tuple is indeed a "DDH tuple". Otherwise, $\mathcal{D}$ outputs 0 to indicated that the challenge tuple is indeed a "random tuple".

Now, we assess the probability that $\mathcal{D}$ output 1. If $(g, h, y_1, y_2)$ is "DDH tuple", then $\mathcal{D}$ provides a perfect simulation for $\mathcal{F}$. It is no hard to see that $\mathcal{D}$ will outputs 1 with probability at least $\epsilon$.

On the other hand, if the challenge tuple is a "random tuple", then with probability $1 - 1/q$ it is not a "DDH tuple". The valid signature $(e^*, s^*)$ implies that $\mathcal{F}$ has successfully attacked the soundness of protocol EDL_MWZ. According to the lemma 2, the probability that $\mathcal{F}$ produces such a valid signature is no more than $\eta$. Thus, in this case, $\mathcal{F}$ outputs a valid signature with probability at most $\eta - \frac{\eta}{q} + \frac{\epsilon}{q} < \eta + \frac{1}{q}$.

Put all together, we can obtain

$$\begin{aligned}
&|\Pr[a, b \longleftarrow \mathbb{Z}_q^* : \mathcal{D}(g, g^a, g^b, g^{ab})] - \Pr[a, b, c \longleftarrow \mathbb{Z}_q^* : \mathcal{D}(g, g^a, g^b, g^c)]| \\
&\geq \epsilon - (\eta + \frac{1}{q}) \\
&\geq \epsilon'
\end{aligned}$$

The running time of $\mathcal{D}$ mainly includes the running time of $\mathcal{F}$, he time of simulating hash function queries, the time of simulating signature queries, and the time of verifying the valid of the forged signature. In very query, a four element multi-exponentiation is required to generate the signature. The cost of this four element multi-exponentiation is about 55% more than the cost of a single exponentiation. Hence, $t' \leq t + 1.55(q_S + q_H + q_G + 1)t_e$. □

## 6 Conclusion

Deffie-Hellman assumptions are well studied mathematical problems in cryptographic field. Nowadays, provable security is a widely accepted necessary requirement for cryptographic primitives. In this paper, we have proposed two new fast signature schemes with tight security reduction to Diffie-Hellman problems. A remarkable advantage of our schemes is its high efficiency: there is a two-exponentiation reduction in our schemes when comparing with their counterpart like *EDL*, CM and KW-DDH schemes. To achieve high efficiency of signature schemes, we proposed a new protocol EDL_MWZ to implement the proof of equality of discrete logarithm. The new protocol outperforms its old counterpart, the Chaum and Pedersen protocol, as two exponentiations is saved by the new protocol. The EDL_MWZ protocol may be of independent interests, as it can accelerate a variety cryptographic schemes such as threshold signature (or decryption) schemes, verifiable secret sharing schemes, etc..

## References

[1] D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, Proceedings of Eurocrypt 2004, volume 3027 of Lecture Notes in Computer Science, pages 56-73. Springer-Verlag, 2004.

[2] Benoit Chevallier-Mames. An efficient CDH-based signature scheme with a tight security reduction. Advances in Cryptology-CRYPTO 2005,Volume 3621 of Lecture Notes in Computer Science, pages 511-526, 2005.

[3] Ernest Brickell, Daniel Gordon, Kevin McCurley, and David Wilson. Fast exponentiation with precomputation. In R.A. Rueppel, editor, Proceedings of Eurocrypt 92, volume 658 of Lecture Notes in Computer Science, pages 200-207. Springer-Verlag, May 1992.

[4] Bodo Moller. Algorithms for Multi-exponentiation. Selected Areas in Cryptography - SAC 2001, volume 2259 of Lecture Notes in Computer Science, pages 165-180, 2001.

[5] M. Bellare, P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. Proceeding of the 1st ACM Conference on Computer and Communications Security, pages 62-73, 1993.

[6] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In U. Maurer, editor, Advances in Cryptology-EUROCRYPT'96, Volume 1070 of Lecture Notes in Computer Science, pages 399-416. Springer-Verlag, 1996.

[7] D. Boneh, B. Lymn and H. Shacham. Short signatures from the Weil pairing. Prodeedings of Asiacrypt 2001, Volume 2248 of Lecture Notes in Computer Science, pages514-532, 2001.

[8] M. Bellare and G. Neven. Multi-signature in the plain pulic-key modle and s genral forking lemma. The 13th ACM Conference on Computer and Communication Security, pages 390-398, 2006,

[9] David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In David Chaum and Wyn Price, editors, Proceedings of Eurocrypt 87, volume 304 of Lecture Notes in Computer Science, pages 127-142. Springer-Verlag, 1987.

[10] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. In Proceedings of the 30th annual ACM symposium on Theory of Computing, pages 209-218. ACM Press, 1998.

[11] D. Chaum and T. Pedersen. Wallet databases with observers. In Advances in Cryptology-Crypto'92, Volume 704, pages 89-105, 1992.

[12] R. Cramer and V. Shoup. Signature scheme based on the strong RSA assumption. ACM Transactions on Information and System Security, 3(3):161-185, 2000.

[13] W. Diffie and M.E. Hellman. New directions in cryptography. IEEE Transactions on Information Theory, IT-22(6):644-654, 1976.

[14] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, IT-31(4):469-472, 1985.

[15] U.Feige, A. Fiat and A. Shamir. Zero-knowledge proofs of Identity. Journal of Cryptology, vol. 1, pages 77-95, 1988.

[16] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A.M. Odlyzko, editor, Advances in Cryptology-CRYPTO'86, volume 263 of Lecture Notes in Computer Science, pages 186-194. Springer-Verlag, 1987.

[17] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In M. Bellare, editor, Advances in Cryptology-EUROCRYPT'99, volume 1592 of Lecture Notes in Computer Science, pages 123-139. Springer-Verlag, 1999.

[18] M. Girault. An identity-based identification scheme based on discrete logarithms modulo a composite number. In Advances in Cryptology-EUROCRYPT'90, volume 473 of Lecture Notes in Computer Science, pages 481-486. Springer-Verlag, 1991.

[19] E. J. Goh and S. Jarecki, A signature scheme as secure as the Diffie-Hellman problem. In E. Biham, editor, Advances in Cryptology EUROCRYPT 2003, Lecture Notes in Computer Science, pages 401-415. Springer-Verlag, 2003

[20] E. J. Goh, S. Jareckiz, J. Katzx N. Wang. Efficient signature schemes with tight reductions to the Diffie-Hellman problems. http://cr.yp.to/bib/2003/katz-sigs.pdf.

[21] S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptive chosen message attacks. SIAM J. Comput. 17, 2, pages 281-308, 1988.

[22] J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. In ACM Conference on Computer and Communications Security, pages 155-164. ACM Press, 2003.

[23] A. Joux and K. Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. In Journal of Cryptology, Volume 16, No. 4, pages 239-247, 2003.

[24] S. Micali and L. Reyzin. Improving the exact security of digital signatre schemes. Journal of Cryptology, 15(1):1-18, 2002.

[25] Ueli Maurer and Stefan Wolf. The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. SIAM Journal on Computing, 28(5):1689-1721, 1999.

[26] National Institute of Standards ans Technology. JIST FIPS PUB 186, Digital signature standard. U.S. Department of Commerce (1994).

[27] K. Ohta and T. Okamoto. A digital multisignature scheme based on the Fiat-Shamir scheme. Advance in Cryptology-ASIACRYPT'91, Volume 739 of Lecture Notes in Computer Science, pages 139-148, Springer-Verlag, 1991.

[28] IEEE P1363. IEEE Standard Specifications for Public-Key Cryptography. IEEE Computer Society, August 2000.

[29] D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. Maurer, editor, Advances in Cryptology-EUROCRYPT'96, volume 1070 of Lecture Notes in Computer Science, pages 387-398. Springer- Verlag, 1996.

[30] R.L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120-126, 1978.

[31] C.-P. Schnorr. Efficient signature generation by smart cards. Journal of Cryptology, 4(3):161-174, 1991.

[32] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, Proceedings of Eurocrypt 97, volume 1233 of Lecture Notes in Computer Science, pages 256-266. Springer-Verlag, 1997.

[33] Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In Joe Killian, editor, Proceedings of Crypto 01, volume 2139 of Lecture Notes in Computer Science, pages 355-367. Springer-Verlag, August 2001.

## 7 Appendix: the Proof of Lemma 1

In order to prove the security of the protocol EDL_MWZ, we follow the approach of Feige, Fiat and Shamir [15], firs proving the completeness, then the soundness and, finally the zero knowledge property. The completeness of our protocol is described as follows.

**Lemma 3 (Completeness).** *If the prover knows* $\mathrm{Dlog}_g y_1 = \mathrm{Dlog}_h y_2 = x$, *then the verifier will accept the proof with probability 1.*

**Proof.** This can be easily argued. Assuming that the prover and the verifier follow the protocol honestly. Let $(v, c, s)$ be the output of an interactive proof between the prover and verifier. Since $x = \mathrm{Dlog}_g y_1 = \mathrm{Dlog}_h y_2$, the verifier computes $(g^n h)^s (y_1^n y_2)^c = (g^n)^{s+xc} h^{s+xc} = (g^n h)^k = v$, which implies that the verifier accepts the proof. Thus, the verifier will accept the proof of an honest prover with probability 1. □

The proof of soundness consists in proving that, if an adversarial prover $\mathcal{A}$ can make its proof accepted but in fact $\mathrm{Dlog}_g y_1 \neq \mathrm{Dlog}_h y_2$, then we can use it as a subroutine to solve the discrete logarithm problem. Unlike existing protocols, he soundness of our protocol is proved in the random oracle model. During the proof, the hash function $H$ is modeled as a random oracle. We argue the soundness by using general fork lemma [8] and rewinding technique.

**Lemma 4 (Soundness).** *Assume* $\mathrm{Dlog}_g y_1 \neq \mathrm{Dlog}_h y_2$. *In the random oracle model, if there exists a prover that makes the verifier accept the proof, then there exists an algorithm* $\mathcal{B}$ *which can solve the discrete logarithm problem.*

**Proof.** Assuming that we have an adversarial prover $\mathcal{A}$ which can cheat (here, the word "cheat" means that the prover is accepted but $\mathrm{Dlog}_g y_1 \neq \mathrm{Dlog}_h y_2$) the verifier with non-negligible probability. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ to solve the discrete logarithm problem. The detailed description of algorithm $\mathcal{B}$ is as follows.

Algorithm $\mathcal{B}$ is given as input an instance $g, h \in \mathbb{G}_{g,q}$ of discrete logarithm problem. Its goal is to output the value $\mathrm{Dlog}_g h$. To this end, $\mathcal{B}$ initializes the adversary $\mathcal{A}$ with the public parameters $g, h$ and $\mathbb{G}_{g,q}$. Then $\mathcal{A}$ selects the public values $y_1$ and $y_2$. Assuming that $\mathrm{Dlog}_g y_1 \neq \mathrm{Dlog}_h y_2$ and $\mathcal{A}$ can interact with $\mathcal{B}$ at most $q_I$ times. $\mathcal{B}$ simulates $\mathcal{A}$'s hash function queries and interacts with $\mathcal{A}$ as follows.

**Hash queries.** To manipulate hash function queries, $\mathcal{B}$ first chooses a random number $n$ from $\mathbb{Z}_q^*$ for $H(g, h, y_1, y_2)$. If $\mathcal{A}$ asked a hash query on the point $(g, h, y_1, y_2)$, the value $n$ is returned. Else, $\mathcal{B}$ checks whether it was asked before. If so, the previously assigned value is returned. Otherwise, $\mathcal{B}$ responses with a random number $n$ chosen from $\mathbb{Z}_q^*$.

**Interaction.** When $\mathcal{A}$ launches a proof interaction, $\mathcal{B}$ responses it with a random number $c$ selected from $\mathbb{Z}_q^*$.

At some interaction, $\mathcal{A}$ passes the proof successfully and outputs the tuple $(v_R, c_R, s_R)$ which is the view of the verifier during this proof.

Now, $\mathcal{B}$ run the forking algorithm [8] $\mathsf{F}_{\mathcal{A}}(X)$ (where $X = (g, h)$), which with probability frk returns $(1, (v_R, c_R, s_R), (v_R, c'_R, s'_R))$ with $c_R \neq c'_R$. These two tuples are such that

$$v_R = (g^n h)^{s_R} (y_1^n y_2)^{c_R} = (g^n h)^{s'_R} (y_1^n y_2)^{c'_R}.$$

Let $w_1 = \frac{s_R - s'_R}{c'_R - c_R}$, we can obtain

$$y_1^n y_2 = (g^n h)^{w_1}. \quad (1)$$

$\mathcal{B}$ rewinds the above game to the point where $\mathcal{A}$ asks for the query $H(g, h, y_1, y_2)$. At this time, $\mathcal{B}$ responses $\mathcal{A}$ with $H(g, h, y_1, y_2) = 2n$. Using the general forking lemma again, we can also obtain

$$y_1^{2n} y_2 = (g^{2n} h)^{w_2}. \quad (2)$$

$\mathcal{B}$ rewinds the above game again to the same point and responses $\mathcal{A}$ with $H(g, h, y_1, y_2) = 3n$. Using the general forking lemma for the third time, we can also obtain

$$y_1^{3n} y_2 = (g^{3n} h)^{w_3}. \quad (3)$$

$\frac{(2)}{(1)}$ and $\frac{(3)}{(2)}$, we can obtain

$$y_1^n = g^{2nw_2 - nw_1} h^{w_2 - w_1} = g^{3nw_3 - 2nw_2} h^{w_3 - w_2}.$$

Obviously, $g^{3nw_3 - 4nw_2 + nw_1} = h^{2w_2 - w_1 - w_3}$. We argue that $2w_2 - w_1 - w_3 \neq 0$. This can be easily done by contradiction. Assume $2w_2 - w_1 - w_3 = 0$, then $3nw_3 - 4nw_2 + nw_1 = 0$. Substituting $2w_2$ with $w_1 + w_3$ in the latter formula, we can obtain $w_1 = w_3$. Hence, $y_1 = g^{w_1}$ and $y_2 = h^{w_1}$, which contradicts with the assumption that $\mathrm{Dlog}_g y_1 \neq \mathrm{Dlog}_h y_2$. So, it is the case that $2w_2 - w_1 - w_3 \neq 0$. Now, we have $h = g^{\frac{3nw_3 - 4nw_2 + nw_1}{2w_2 - w_1 - w_3}}$. This completes the proof. $\square$

**Lemma 5 (Zero knowledge).** *The protocol is honest verifier statistically zero knowledge.*

**Proof.** We first remind that the zero knowledge property is verified if the view of the honest verifier interacting with the honest prover can be simulated by a polynomial time algorithm (which is also called the simulator) with only the public parameter as input. A protocol is said to statistically zero knowledge if the view and the output of the simulator are statistically indistinguishable.

We now construct the simulator $\mathcal{S}$ as: when given the public parameters $g, h, y_1, y_2$, $\mathcal{S}$ chooses random numbers $c, s$ from $\mathbb{Z}_q^*$ and computes $v = (g^n h)^s (y_1^n y_2)^c$, where $n = H(g, h, y_1, y_2)$. It is no hard to verify that the distribution of the transcript $(v, c, s)$ is identically to the distribution of the view of the honest verifier interacting with the prover (who knows the secret information $x$). Thus, we have done this lemma. $\square$