Algorithmica (2012) 64:246–266
DOI 10.1007/s00453-011-9593-3

Zero-Knowledge Argument for Simultaneous Discrete Logarithms

Sherman S.M. Chow · Changshe Ma · Jian Weng

Received: 15 July 2010 / Accepted: 2 November 2011 / Published online: 29 November 2011

Abstract In Crypto 1992, Chaum and Pedersen introduced a protocol (CP protocol for short) for proving the equality of two discrete logarithms (EQDL) with unconditional soundness, which is widely used nowadays and plays a central role in DL-based cryptography. Somewhat surprisingly, the CP protocol has never been improved for nearly two decades since its advent. We note that the CP protocol is usually used as a non-interactive proof by using the Fiat-Shamir heuristic, which inevitably relies on the random oracle model (ROM) and assumes that the adversary is computationally

This is the full version of [10] which appeared in the 16th International Conference on Computing and Combinatorics (COCOON'10).

S.S.M. Chow
Department of Combinatorics and Optimization, and Centre for Applied Cryptographic Research, University of Waterloo, Waterloo, Ontario, Canada, N2L3G1
e-mail: smchow@math.uwaterloo.ca

C. Ma
School of Computer, South China Normal University, Guangzhou, 510631, China
e-mail: changshema@gmail.com

J. Weng (✉)
Department of Computer Science, Jinan University, Guangzhou 510632, China
e-mail: cryptjweng@gmail.com

J. Weng
State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China

J. Weng
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

J. Weng
Emergency Technology Research Center of Risk Evaluation and Prewarning on Public Network Security, Guangdong 510632, China

bounded. In this paper, we present an EQDL protocol in the ROM which saves approximately 40% of the computational cost and approximately 33% of the prover's outgoing message size when instantiated with the same security parameter. The catch is that our security guarantee only holds for computationally bounded adversaries. Our idea can be naturally extended for simultaneously showing the equality of n discrete logarithms with $O(1)$-size commitment, in contrast to the n-element adaption of the CP protocol which requires $O(n)$-size. This improvement benefits a variety of interesting cryptosystems, ranging from signatures and anonymous credential systems, to verifiable secret sharing and threshold cryptosystems. As an example, we present a signature scheme that only takes one (offline) exponentiation to sign, without utilizing pairing, relying on the standard decisional Diffie-Hellman assumption.

Keywords Equality of discrete logarithms · Simultaneous discrete logarithms · Honest-verifier zero-knowledge argument · Online/offline signature · Diffie-Hellman problem

1 Introduction

The protocol for proving the equality of two discrete logarithms plays a central role in cryptography. Such a protocol, executed by a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, can be roughly explained as follows: Suppose $\mathbb{G}$ is a cyclic group with prime order q, and g and h are two random elements chosen from $\mathbb{G}$. $\mathcal{P}$ knows $x \in \mathbb{Z}_q$ such that $y_1 = g^x$ and $y_2 = h^x$. After executing the protocol, $\mathcal{P}$ must convince $\mathcal{V}$ that y_1 and y_2 indeed have the same exponent with respect to the base g and h respectively, i.e., $\log_g y_1 = \log_h y_2$. The idea can be naturally extended to the *simultaneous discrete logarithm* problem, where one wants to show that $y_0 = g_0^x, y_1 = g_1^x, \ldots, y_{n-1} = g_{n-1}^x$ when given n generators $g_0, g_1, \ldots, g_{n-1}$ of group $\mathbb{G}$.

Chaum and Pedersen [7] introduced a famous protocol (hereinafter referred as CP protocol) for proving the equality of two discrete logarithms, which has been widely used to design a variety of cryptosystems. Examples include but not limited to digital signatures [8, 15], verifiable secret sharing [25], threshold cryptosystems [23, 24], fair exchange protocols [1], pseudonym [18], electronic payment [6], electronic voting [9, 11], etc. Any improvement on the CP protocol is influential since it benefits all these cryptosystems accordingly.

Somewhat surprisingly, the CP protocol has never been improved for nearly two decades since its advent. A natural question to ask is whether we can improve the CP protocol. In this paper, we answer the question motivating the above question instead—can we have a more efficient protocol which "benefits" the cryptosystems utilizing the CP protocol?

To answer our question, we investigate how the CP protocol is typically used and what kinds of its properties have been utilized in cryptographic or security applications. In many cases, interactive proof is undesirable, if not impossible. Examples include digital signatures which provide non-repudiation property, and multi-signatures [3, 19] which require all signers to be online if it is done in an interactive setting. For these applications, the CP protocol is turned into a non-interactive proof by the Fiat-Shamir heuristic [13], where the prover "challenges" herself by treating the output of

the hash function as the challenge given by the verifier. This inevitably relies on the random oracle model (ROM) [4]. On the other hand, the CP protocol, when used as an interactive proof, can be proven secure without relying on the ROM.

As a zero-knowledge proof system, the CP protocol provides unconditional soundness. Roughly speaking, this (soundness) means the prover cannot cheat without being caught in most cases, without any condition imposed on the prover, in particular, the prover may possess unlimited computational power. For real-world applications, we may feel comfortable to assume that the computational ability of an adversary is bounded, and hence it is sufficient to use a zero-knowledge *argument*, a system with computational soundness.

1.1 Our Contributions

Based on the above two simple yet important observations, we propose a protocol which gives a zero-knowledge argument system for simultaneous discrete logarithm. In the CP protocol, the prover $\mathcal{P}$ firstly generates two group elements as the commitments. We carefully integrate these two elements together in our protocol, which gives a higher performance in terms of both computational cost and communication overhead. More concretely, our protocol offers about 40% computational cost saving, and saves one group element from the communication overhead, when instantiated with the same security parameter. We remark that it is not entirely trivial to combine two commitments into a single element. A class of trial attempts is insecure [10].

Our protocol can be naturally extended for simultaneous discrete logarithm for n elements where n is an integer greater than 2. Compared with the n-element adaption of the CP protocol which uses $O(n)$ commitments, we achieve $O(1)$ size.

Finally, we use our protocol to construct two signature schemes. Both schemes are very efficient (refer to Table 2). In particular, we give a signature scheme that only takes one offline exponentiation and no online exponentiation to sign, under the standard decisional Diffie-Hellman assumption, without utilizing pairing.

1.2 Organization

Section 2 reviews some preliminaries, including three problems related to discrete logarithm, the definition and the security notions for zero-knowledge protocol and signature schemes. In Sect. 3, we present our new protocol for showing the equality of two discrete logarithms, and its extension for n simultaneous discrete logarithms. In Sect. 4, we use our protocol to construct two efficient signature schemes. Finally, Sect. 5 concludes this paper.

2 Preliminaries

2.1 Notations and Complexity Problems

We explain some notations used in the rest of this paper. For a prime q, $\mathbb{Z}_q$ denotes the set $\{0, 1, 2, \ldots, q-1\}$, and $\mathbb{Z}_q^*$ denotes $\mathbb{Z}_q \backslash \{0\}$. For a finite set S, $x \in_R S$ means

choosing an element x uniformly at random from S. We use $\mathbb{G}$ to denote a cyclic group with prime order q and $\ell_q = |\mathbb{G}|$ to denote the bit-length of the binary representation of an element in $\mathbb{G}$.

Definition 1 The discrete logarithm (DL) problem is, given $g, y \in \mathbb{G}$, to compute $x \in \mathbb{Z}_q$ such that $y = g^x$. We say that an algorithm $\mathcal{B}$ can solve the DL problem in group $\mathbb{G}$ with advantage ε if $\Pr[\mathcal{B}(g, y) = x | g \in_R \mathbb{G}, x \in_R \mathbb{Z}_q, y = g^x] \geqslant \varepsilon$.

Definition 2 The computational Diffie-Hellman (CDH) problem is, given $g, g^a, g^b \in \mathbb{G}$ with unknown $a, b \in \mathbb{Z}_q$, to compute g^{ab}. We say that an algorithm $\mathcal{B}$ has advantage ε in solving CDH problem in group $\mathbb{G}$, if $\Pr[\mathcal{B}(g, g^a, g^b) = g^{ab} \big| g \in_R \mathbb{G}, a, b \in_R \mathbb{Z}_q] \geqslant \varepsilon$.

Definition 3 The decisional Diffie-Hellman (DDH) problem is, given g, g^a, g^b, g^c, where $a, b \in_R \mathbb{Z}_q^*$, to tell if $c = ab$ or $c \in_R \mathbb{Z}_q^*$. A tuple (g, g^a, g^b, g^c) is called a "DDH tuple" if $ab = c \mod q$ holds. We say that an algorithm $\mathcal{B}$ has advantage ε in solving the DDH problem in $\mathbb{G}$ if $|\Pr[\mathcal{B}(g, g^a, g^b, g^{ab}) = 1 | g \in_R \mathbb{G}, a, b \in_R \mathbb{Z}_q, c = ab] - \Pr[\mathcal{B}(g, g^a, g^b, g^c) = 1 | g \in_R \mathbb{G}, a, b, c \in_R \mathbb{Z}_q]| \geqslant \varepsilon$.

We say that the ε-DL/CDH/DDH assumption holds if no probabilistic polynomial time (PPT) algorithm can solve the respective problem with an advantage at least ε.

2.2 Zero-Knowledge Argument for Equality of Discrete Logarithms

An interactive argument consists of a common reference string (CRS) generator algorithm $\mathcal{G}$, and a protocol $(\mathcal{P}, \mathcal{V})$ between a prover and a verifier. The CRS generation algorithm $\mathcal{G}$ takes as input a security parameter 1^ℓ, outputs a CRS param and any necessary description of the relation R (e.g., parameters related to the statements to be proven). All these are the public input for the protocol executed between $\mathcal{P}$ and $\mathcal{V}$. In particular, for equality of two discrete logarithms, the description of the relation R will contain two group elements g and h, yet the group elements y_1 and y_2 are up to the choice of $\mathcal{P}$. As a result of the protocol, the verifier $\mathcal{V}$ accepts (outputs $\top$) or rejects (outputs $\bot$) depending on whether the argument provided by $\mathcal{P}$ is acceptable or not. The prover has no local output.

The security notions for honest-verifier zero-knowledge protocols for proving the equality of two logarithms are reviewed as follows.

Definition 4 A protocol for proving the equality of two discrete logarithms is said to be secure, if it simultaneously satisfies the following three properties:

- **Completeness**. For any honest prover $\mathcal{P}$, an honest verifier $\mathcal{V}$ accepts with overwhelming probability.
- **Soundness**. An algorithm $\mathcal{A}$ can ε-break the soundness of the protocol if $\mathcal{A}$ can cheat, without conducting any active attacks, an honest verifier $\mathcal{V}$ (i.e., $\mathcal{A}$ can make $\mathcal{V}$ to accept its proof whereas $\log_g y_1 \neq \log_h y_2$) with at least success probability ε, where the probability is taken over the random coins consumed by $\mathcal{A}$ and $\mathcal{V}$. The

soundness for the protocol means that no PPT adversary can ε-break it. Formally, for all nonuniform PPT adversaries $\mathcal{A}$, and all integers ℓ, we have

$$\begin{aligned}\Pr[(\mathsf{param}, \mathbb{G}, g, h) \leftarrow \mathcal{G}(1^\ell); (y_1, y_2) \leftarrow \mathcal{A} :\\ \mathcal{V} \text{ accepts in } (\mathcal{A}, \mathcal{V})(\mathsf{param}, g, h, y_1, y_2) \wedge\\ \log_g y_1 \neq \log_h y_2\ ] \leq \mathsf{negl}(\ell).\end{aligned}$$

- **(Honest-Verifier) Zero-Knowledge**. We say that a protocol is statistical/computational zero-knowledge, if for every PPT (honest) verifier $\mathcal{V}$, there exists a PPT machine $\mathcal{S}$ (also called simulator), which is not allowed to interact with the prover $\mathcal{P}$, such that the following two distributions are statistically/computationally indistinguishable, when $\log_g y_1 = \log_h y_2$:
  - the output of $\mathcal{V}$ after interacting with a prover $\mathcal{P}$ on the common input of public parameters $(\mathbb{G}, g, h, y_1, y_2)$, and
  - the output of $\mathcal{S}$ which can produce the public parameters $(\mathbb{G}, g, h, y_1, y_2)$.

  Formally, for all integers ℓ, and $\forall y_1 \in \mathbb{G}, y_2 \in \mathbb{G}$, we have the following two distributions statistically/computationally indistinguishable,

  $$\begin{aligned}&\mathrm{view}_{\mathcal{V}}[\mathcal{V}(\mathsf{param}, \mathbb{G}, g, h, y_1, y_2) \leftrightarrow \mathcal{P}(\mathsf{param}, g, h, y_1, y_2) : \mathsf{param} \leftarrow \mathcal{G}(1^\ell)],\\ &\mathcal{S}(\mathsf{param}, s, g, h, y_1, y_2, \text{“}\texttt{setup}\text{”}) : (\mathsf{param}, s) \leftarrow \mathcal{S}(1^\ell, \text{“}\texttt{prove}\text{”})\end{aligned}$$

  where $\mathrm{view}_{\mathcal{V}}$ is the “view” of the verifier in the interaction with $\mathcal{P}$, consisting of its input $(\mathsf{param}, \mathbb{G}, g, h, y_1, y_2)$, its own random coins, and the messages produced by $\mathcal{P}$; and $\mathcal{S}$ is a multi-stage algorithm which either outputs a simulated parameter param and its “trapdoor” information s when taken an input of a security parameter 1^ℓ and the string “`setup`”, or outputs a simulated view when its input is the trapdoor s and the string “`prove`”.

Remark In our protocols, the public parameters should also contain the description of the hash function H. Our proof for soundness requires modeling H as a random oracle.

2.3 Digital Signature and its Existential Unforgeability

A signature scheme is defined by the following triple of algorithms:

- $\mathsf{KeyGen}(1^\ell)$: it takes a security parameter 1^ℓ and outputs a private/public key pair (sk, pk).
- $\mathsf{Sign}(sk, m)$: it outputs a signature σ taking a signing key sk and a message m as inputs.
- $\mathsf{Verify}(pk, \sigma, m)$: this verification algorithm takes as input a public key pk, a signature σ and a message m. It outputs 1 if and only if σ is a valid signature on m under pk, 0 otherwise.

Security is modeled by the existential unforgeability under adaptive chosen message attack (EUF-CMA), defined via the following game between a forger $\mathcal{F}$ and a challenger $\mathcal{C}$:

Setup. The challenger $\mathcal{C}$ runs algorithm $\mathsf{KeyGen}(1^\ell)$ to generate a key pair (pk, sk). It then forwards pk to $\mathcal{F}$, keeping sk to himself.
Signing queries. For each query, $\mathcal{F}$ adaptively issues a message m, the challenger runs $\mathsf{Sign}(sk, m)$ to get a signature σ, which is returned to $\mathcal{F}$.
Forge. $\mathcal{F}$ outputs a message/signature pair (m^*, σ^*) where $\mathsf{Verify}(pk, \sigma^*, m^*) = 1$.

$\mathcal{F}$ wins if m^* did not appear in the signing queries. $\mathcal{F}$'s advantage is defined by $\mathrm{Adv} = \Pr[\mathcal{F}\text{ wins}]$, where the probability is taken over the random coins consumed by $\mathcal{C}$ and $\mathcal{F}$.

Definition 5 We say that a signature scheme is (q_s, ε)-$\mathsf{EUF\text{-}CMA}$ secure, if for any PPT forger $\mathcal{F}$ who asking at most q_s signing queries, his advantage is less than ε.

3 Our Zero-Knowledge Argument System for Simultaneous Discrete Logarithms

3.1 Review of Chaum-Pedersen Protocol

Let $\mathbb{G}$ be a cyclic group of prime order q, and let g, h be two random generators of $\mathbb{G}$. The prover $\mathcal{P}$ picks $x \in \mathbb{Z}_q$, and publishes $y_1 = g^x$ and $y_2 = h^x$. To convince the verifier $\mathcal{V}$ that $\log_g y_1 = \log_h y_2$, the CP protocol [7] proceeds as in Fig. 1. It is well known that the CP protocol is an honest-verifier zero-knowledge protocol.

3.2 Proposed Protocol

Let $H : \mathbb{G}^2 \to \mathbb{Z}_q^*$ be a cryptographic hash function (in the proof of soundness, we shall model it as a random oracle). We define the commitment as $v = (g^z h)^k$ where $z = H(y_1, y_2)$. We assume that the discrete logarithm $\log_g h$ is unknown to all participants. This can be easily ensured in practice by requiring h to be the output of a hash function (which again is modeled as a random oracle in the security proof) with a public seed and a public input. For example, one may set h to be $PRF_s(g)$ where PRF is a pseudorandom function taking the public seed s. Detailed protocol is shown in Fig. 2.

3.3 Security

The security of our protocol can be ensured by the following theorem, which follows directly from Lemmas 1, 2 and 4 given later.

1. $\mathcal{P}$ picks $k \in_R \mathbb{Z}_q$ and sends the pair of commitments $(a_1, a_2) = (g^k, h^k)$ to $\mathcal{V}$.
2. Upon receiving (a_1, a_2), $\mathcal{V}$ sends the challenge $c \in \mathbb{Z}_q^*$ to $\mathcal{P}$.
3. $\mathcal{P}$ sends the response $s = k - cx \mod q$ to $\mathcal{V}$.
4. $\mathcal{V}$ accepts if and only if both $a_1 = g^s y_1^c$ and $a_2 = h^s y_2^c$ hold.

Fig. 1 Chaum-Pedersen protocol for proving $y_1 = g^x \wedge y_2 = h^x$

1. $\mathcal{P}$ picks $k \in_R \mathbb{Z}_q^*$ and sends the commitment $v = (g^z h)^k$ to $\mathcal{V}$, where $z = H(y_1, y_2)$.
2. Upon receiving v, $\mathcal{V}$ sends the challenge $c \in_R \mathbb{Z}_q^*$ to $\mathcal{P}$.
3. $\mathcal{P}$ sends the response $s = k - cx \mod q$ to $\mathcal{V}$.
4. $\mathcal{V}$ accepts if and only if $v = (g^z h)^s (y_1^z y_2)^c$ holds, where $z = H(y_1, y_2)$.

Fig. 2 Our protocol for the statement $y_1 = g^x \wedge y_2 = h^x$

Theorem 1 *In the random oracle model, our protocol is secure according to Definition 4, under the DL assumption in group* $\mathbb{G}$.

3.3.1 Completeness

Lemma 1 *For every honest prover, an honest verifier will accept with probability* 1 *in our protocol.*

Proof Let (v, c, s) be the communication transcript between a prover and a verifier. $x = \log_g y_1 = \log_h y_2$ implies the verification equation always holds as shown below: $(g^z h)^s (y_1^z y_2)^c = (g^z h)^{k-cx} (g^{xn} h^x)^c = (g^z h)^{k-cx+xc} = (g^z h)^k = v$. □

3.3.2 Honest-Verifier Zero-Knowledge

Lemma 2 *Our protocol is honest-verifier statistically zero-knowledge.*

Proof Recall that the zero-knowledge property is satisfied if there exists a polynomial time simulator which can simulate the view of an honest verifier interacting with an honest prover. We now construct this simulator $\mathcal{S}$ as follow: when given the public parameters g, h, y_1, y_2, $\mathcal{S}$ chooses random numbers c, s from $\mathbb{Z}_q^*$ and computes $v = (g^z h)^s (y_1^z y_2)^c$, where $z = H(y_1, y_2)$. It is easy to see that the distribution of the transcript (v, c, s) is identically to the distribution of the view of an honest verifier interacting with the prover (who knows the secret information $\log_g y_1 = \log_h y_2$). □

3.3.3 Soundness

Our security proof for the soundness uses the multiple-forking lemma of Boldyreva, Palacio and Warinschi [5], which is a generalization of the forking lemma proposed by Pointcheval and Stern [22]. In a high level, the forking lemma states that if an algorithm produces an output that has some property on inputs drawn from some distribution, then it is possible to re-run this algorithm and get a new output having the same property, on new inputs but with the same random tape.

Putting this into our perspective, the algorithm producing an output is our adversary, and the property this output possesses is that for the same y_1 and y_2, the adversary can successfully "cheat" which means the output will lead to an accepting transcript. Then our simulator will re-run this adversary but this time we will change the random oracle replies multiple times. Yet, with the probability guaranteed by the lemma, the adversary can still output an accepting transcript even after such changes.

Lemma 3 (Multiple-Forking Lemma) *Fix $\alpha \in \mathbb{Z}^+$ and a set $\mathbb{S}$ such that $|\mathbb{S}| \geq 2$. Let $\mathcal{Y}$ be a randomized algorithm that on input x and a sequence of elements $s_1, \ldots, s_\alpha \in \mathbb{S}$, returns a triple (I, J, σ) where $0 \leq J \leq I \leq \alpha$ and a string σ. Let $\mathcal{G}$ be a randomized algorithm that takes no input and returns a string. Finally, define the multiple-forking algorithm $\mathcal{F}_{\mathcal{Y},n}$ associated to $\mathcal{Y}$ and an odd integer $n \geq 1$ as in Fig. 3, and*

$$\begin{aligned}\mathsf{acc} &= \Pr[x \xleftarrow{\$} \mathcal{G}; s_1, \ldots, s_\alpha \xleftarrow{\$} \mathbb{S}; (I, J, \sigma^{(0)}) \leftarrow \mathcal{Y}(x, s_1, \ldots, s_\alpha; \rho) : \\ &\qquad I \geq 1 \wedge J \geq 1], \\ \mathsf{frk} &= \Pr[x \xleftarrow{\$} \mathcal{G}; \mathcal{F}_{\mathcal{Y},n}(x) \neq \perp].\end{aligned}$$

Then

$$\mathsf{frk} \geq \mathsf{acc}\left(\frac{\mathsf{acc}^n}{\alpha^{2n}} - \frac{n}{|\mathbb{S}|}\right), \quad \text{i.e.,} \quad \mathsf{acc} \leq \sqrt[n+1]{\alpha^{2n}\mathsf{frk}} + \sqrt[n+1]{\frac{n\alpha^{2n}}{|\mathbb{S}|}}.$$

The multiple-forking algorithm $\mathcal{F}_{\mathcal{Y},n}$ associated to $\mathcal{Y}$ as defined Fig. 3 forms a major part of our simulator. A forking here corresponds to changing a particular value of s's. The number of such forkings is governed by n. Now we are ready to consider the soundness of our protocol.

Lemma 4 (Soundness) *In the random oracle model (the hash function H will be modeled as a random oracle), if there exists an adversary $\mathcal{A}$ that can ε-break the soundness of our protocol (i.e. $\mathcal{V}$ accepts but $\log_g y_1 \neq \log_h y_2$), there exists an algorithm $\mathcal{B}$ which can ε^*-solve the DL problem with*

$$\varepsilon^* \geq \mathsf{frk} \geq \varepsilon\left(\frac{\varepsilon^5}{(Q_H Q_C)^{10}} - \frac{5}{|\mathbb{Z}_q^*|}\right),$$

where Q_H is the number of random oracle query made by $\mathcal{A}$ and Q_C is the number of interactions between $\mathcal{A}$ and $\mathcal{B}$.

Proof Assume that there exists an adversarial prover $\mathcal{A}$ who can cheat with a non-negligible probability at least ε within time t. We show how to construct an algorithm $\mathcal{B}$ which can solve the DL problem (g, h) (i.e. output $\log_g h$) with probability at least ε^* within time t^*. $\mathcal{B}$ first gives g and h to $\mathcal{A}$ as the group elements used in the public parameters, and simulates the random oracle $H(\cdot, \cdot)$ in standard way. In more details, $\mathcal{B}$ maintains a list L_H contains the queried values together with the answers given to $\mathcal{A}$. For every query $\mathcal{A}$ makes, if L_H does not contain it, a random answer will be generated by $\mathcal{B}$ and returned to $\mathcal{A}$. Otherwise, the answer will be retrieved from L_H, i.e., $\mathcal{B}$ simulates the hash function queries by picking $z \in_R \mathbb{Z}_q^*$, stores it with the query and returns z consistently.

Essentially, $\mathcal{B}$ gives $\mathcal{A}$ the public parameters $\mathsf{param} = (g, h, H(\cdot, \cdot))$. Now $\mathcal{A}$ generates the values y_1 and y_2 it wants to prove about. $\mathcal{A}$'s goal is to cheat the verifier $\mathcal{V}$ to accept so we can assume that $\log_g y_1 \neq \log_h y_2$. $\mathcal{B}$ also returns random challenges $c \in_R \mathbb{Z}_q^*$ to $\mathcal{A}$. Instead of directly interacting with $\mathcal{A}$, $\mathcal{B}$ runs the algorithm

Pick random coins ρ for $\mathcal{Y}$
$s_1, \ldots, s_\alpha \xleftarrow{\$} \mathbb{S}$; $(I, J, \sigma^{(0)}) \leftarrow \mathcal{Y}(x, s_1, \ldots, s_\alpha; \rho)$;
if $(I = 0) \vee (J = 0)$ **then**
  return $\perp$;
else
  $i \leftarrow 1$;
  $s_I^{(i)}, \ldots, s_\alpha^{(i)} \xleftarrow{\$} \mathbb{S}$; $(I^{(i)}, J^{(i)}, \sigma^{(i)}) \leftarrow \mathcal{Y}(x, s_1, \ldots, s_{I-1}, s_I^{(i)}, \ldots, s_\alpha^{(i)}; \rho)$;
  if $((I^{(i)}, J^{(i)}) \neq (I, J)) \vee (s_I^{(i)} = s_I)$ **then**
    return $\perp$;
  end if
  $i \leftarrow 2$;
end if
while $i < n$ **do**
  $s_J^{(i)}, \ldots, s_\alpha^{(i)} \xleftarrow{\$} \mathbb{S}$; $(I^{(i)}, J^{(i)}, \sigma^{(i)}) \leftarrow \mathcal{Y}(x, s_1, \ldots, s_{J-1}, s_J^{(i)}, \ldots, s_\alpha^{(i)}; \rho)$;
  if $((I^{(i)}, J^{(i)}) \neq (I, J)) \vee (s_J^{(i)} = s_J^{(i-1)})$ **then**
    return $\perp$;
  end if
  $i \leftarrow i + 1$;
  $s_I^{(i)}, \ldots, s_\alpha^{(i)} \xleftarrow{\$} \mathbb{S}$; $(I^{(i)}, J^{(i)}, \sigma^{(i)}) \leftarrow \mathcal{Y}(x, s_1, \ldots, s_{J-1}, s_J^{(i-1)}, \ldots, s_{I-1}^{(i-1)}, s_I^{(i)}, \ldots, s_\alpha^{(i)}; \rho)$;
  if $((I^{(i)}, J^{(i)}) \neq (I, J)) \vee (s_I^{(i)} = s_I^{(i-1)})$ **then**
    return $\perp$;
  end if
  $i \leftarrow i + 1$;
end while
return $\sigma^{(0)}, \ldots, \sigma^{(n)}$;

Fig. 3 Multiple-forking algorithm

$\mathcal{F}_{\mathcal{Y},5}(\mathsf{param})$ as described in the multiple-forking lemma. The algorithm $\mathcal{Y}$ here is a wrapper that takes as explicit input the answers from the random oracle $H(\cdot, \cdot)$ and the random challenges given by $\mathcal{B}$, calls $\mathcal{A}$ and returns its output together with two integers I, J. One of the integers is the index of $\mathcal{A}$'s calls to the random oracle $H(\cdot, \cdot)$ and the other is the index of the challenge corresponding to the cheat given by $\mathcal{A}$.

By running the multiple-forking algorithm, either $\mathcal{B}$ aborts prematurely (when $\mathcal{F}_{\mathcal{Y},5}(\mathsf{param})$ returns $\perp$), or gets six "cheating" transcripts (the output of the interaction between $\mathcal{P}$ and $\mathcal{V}$ in the real protocol which corresponds to the interaction between $\mathcal{A}$ and $\mathcal{B}$ in the proof) in the form of $\{(v_1, c_1^{(i)}, s_1^{(i)}), (v_1, c_2^{(i)}, s_2^{(i)})\}$ for $i = 1, 2, 3$. As all these transcripts correspond to valid proofs, we have the following equation hold in particular,

$$v_1 = (g^{z_1} h)^{s_1^{(1)}} (y_1^{z_1} y_2)^{c_1^{(1)}} = (g^{z_1} h)^{s_2^{(1)}} (y_1^{z_1} y_2)^{c_2^{(1)}},$$

where z_1 is the random oracle reply for $H(y_1, y_2)$ for $i = 1$.

Let $w_1 = \frac{s_1^{(1)} - s_2^{(1)}}{c_2^{(1)} - c_1^{(1)}} \mod q$, then we can obtain

$$y_1^{z_1} y_2 = (g^{z_1} h)^{w_1}. \tag{1}$$

Similarly, for $i = 2, 3$, we can also obtain the following two equations:

$$y_1^{z_2} y_2 = (g^{z_2} h)^{w_2}, \tag{2}$$

$$y_1^{z_3} y_2 = (g^{z_3} h)^{w_3}, \tag{3}$$

where $w_i = \frac{s_1^{(i)} - s_2^{(i)}}{c_2^{(i)} - c_1^{(i)}} \mod q$, z_i is the random oracle reply for $H(y_1, y_2)$, for $i = 2, 3$, and all of z_1, z_2, z_3 are distinct.

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

$$y_1^{(z_1 - z_2)} = g^{(z_1 w_1 - z_2 w_2)} h^{(w_1 - w_2)}, \tag{4}$$

$$y_1^{(z_1 - z_3)} = g^{(z_1 w_1 - z_3 w_3)} h^{(w_1 - w_3)}. \tag{5}$$

According to the probing strategy, z_1, z_2, z_3 are all distinct. From (4) and (5),

$$g^{(\frac{z_1 w_1 - z_2 w_2}{z_1 - z_2} - \frac{z_1 w_1 - z_3 w_3}{z_1 - z_3})} = h^{(\frac{w_1 - w_3}{z_1 - z_3} - \frac{w_1 - w_2}{z_1 - z_2})}. \tag{6}$$

The solution of the DL problem instance is

$$\left(\frac{z_1 w_1 - z_2 w_2}{z_1 - z_2} - \frac{z_1 w_1 - z_3 w_3}{z_1 - z_3}\right) \Big/ \left(\frac{w_1 - w_3}{z_1 - z_3} - \frac{w_1 - w_2}{z_1 - z_2}\right) \mod q.$$

We remain to argue that $\frac{w_1 - w_3}{z_1 - z_3} - \frac{w_1 - w_2}{z_1 - z_2} \neq 0 \mod q$. Assume to the contrary that

$$\frac{w_1 - w_3}{z_1 - z_3} - \frac{w_1 - w_2}{z_1 - z_2} = 0 \mod q, \tag{7}$$

since we are working in a prime-order group, considering the left hand side of (6), we have

$$\frac{z_1 w_1 - z_2 w_2}{z_1 - z_2} - \frac{z_1 w_1 - z_3 w_3}{z_1 - z_3} = 0 \mod q. \tag{8}$$

Consider $z_1 \times (7) + (8)$,

$$z_1 \left(\frac{w_1 - w_3}{z_1 - z_3} - \frac{w_1 - w_2}{z_1 - z_2}\right) + \left(\frac{z_1 w_1 - z_2 w_2}{z_1 - z_2} - \frac{z_1 w_1 - z_3 w_3}{z_1 - z_3}\right)$$

$$= z_1(0) + 0 \mod q,$$

$$\frac{z_1 w_1 - z_1 w_3}{z_1 - z_3} - \frac{z_1 w_1 - z_1 w_2}{z_1 - z_2} + \frac{z_1 w_1 - z_2 w_2}{z_1 - z_2} - \frac{z_1 w_1 - z_3 w_3}{z_1 - z_3} = 0 \mod q,$$

$$\frac{z_1 w_1 - z_1 w_3}{z_1 - z_3} - \frac{z_1 w_1 - z_3 w_3}{z_1 - z_3} - \frac{z_1 w_1 - z_1 w_2}{z_1 - z_2} + \frac{z_1 w_1 - z_2 w_2}{z_1 - z_2} = 0 \mod q,$$

$$\frac{(-z_1 + z_3) w_3}{z_1 - z_3} + \frac{(z_1 - z_2) w_2}{z_1 - z_2} = 0 \mod q$$

$$w_2 = w_3 \mod q.$$

1. $\mathcal{P}$ picks $k \in_R \mathbb{Z}_q^*$ and sends the commitment $v = (g_0 g_1^{z_1} \cdots g_{n-1}^{z_{n-1}})^k$ to $\mathcal{V}$, where $z_i = H(i, y_1, \ldots, y_{n-1})$ for $i \in [1, n-1]$.
2. Upon receiving v, $\mathcal{V}$ sends the challenge $c \in_R \mathbb{Z}_q^*$ to $\mathcal{P}$.
3. $\mathcal{P}$ sends the response $s = k - cx \mod q$ to $\mathcal{V}$.
4. $\mathcal{V}$ accepts if and only if $v = (g_0 g_1^{z_1} \cdots g_{n-1}^{z_{n-1}})^s (y_0 y_1^{z_1} \cdots y_{n-1}^{z_{n-1}})^c$ holds, where $z_i = H(i, y_1, \ldots, y_{n-1})$ for $i \in [1, n-1]$.

Fig. 4 Our protocol for proving $y_0 = g_0^x, \ldots, y_{n-1} = g_{n-1}^x$

Consider $\frac{(2)}{(3)}$ with w_3 substituted by w_2, we have $y_1^{z_2 - z_3} = g^{(z_2 - z_3)(w_2)}$. Recall that $z_2 \neq z_3$, we thus have $y_1 = g^{w_2}$. Substitute this y_1 back to (3), we get $y_2 = h^{w_2}$. This contradicts the premise $\log_g y_1 \neq \log_h y_2$, and hence we have proven that the solution of the DL problem instance is valid.

The success probability bound is given by the multiple-forking lemma with $n = 5$. □

3.4 Extension to n-Element

Our protocol presented in Fig. 2 can be naturally extended to show the equality of n discrete logarithms, i.e., $\log_{g_0} y_0 = \log_{g_1} y_1 = \cdots = \log_{g_{n-1}} y_{n-1} = x$. Extended protocol is given in Fig. 4.

Now we discuss how to modify the proof of Theorem 1 to fit for our extended protocol. Our protocol can be seen as borrowing some techniques from the multi-signature scheme in [3].

Solving the DL problem for n-element case For a DL problem instance $g, h \in \mathbb{G}^2$, $\mathcal{B}$ sets $g_{l^*} = g$ and $g_0 = h$, and picks random $g_i \in_R \mathbb{G}$ for $l \in \{1, \ldots, n\} \backslash \{l^*\}$, for simplicity. $\mathcal{B}$ then uses basically the same strategy when (indirectly) interacting with $\mathcal{A}$.

In the first round ($i = 1$) before any forking, we obtain

$$\begin{aligned} v_1 &= (g_0 g_1^{z_1^{(1)}} \cdots g_n^{z_n^{(1)}})^{s_1^{(1)}} (y_0 y_1^{z_1^{(1)}} \cdots y_n^{z_n^{(1)}})^{c_1^{(1)}} \\ &= (g_0 g_1^{z_1^{(1)}} \cdots g_n^{z_n^{(1)}})^{s_2^{(1)}} (y_0 y_1^{z_1^{(1)}} \cdots y_n^{z_n^{(1)}})^{c_2^{(1)}}, \end{aligned}$$

where $(v_1, c_1^{(1)}, s_1^{(1)})$ and $(v_1, c_2^{(1)}, s_2^{(1)})$ are the outputs of the interaction between $\mathcal{P}$ and $\mathcal{V}$.

Let $w_1 = \frac{s_1^{(1)} - s_2^{(1)}}{c_2^{(1)} - c_1^{(1)}} \mod q$, then we can obtain

$$(g_0 g_1^{z_1^{(1)}} \cdots g_{n-1}^{z_{n-1}^{(1)}})^{w_1} = (y_0 y_1^{z_1^{(1)}} \cdots y_{n-1}^{z_{n-1}^{(1)}}). \tag{9}$$

Similarly, for $i = 2, 3$, we can also obtain the following two equations:

$$(g_0 g_1^{z_1^{(2)}} \cdots g_{n-1}^{z_{n-1}^{(2)}})^{w_2} = (y_0 y_1^{z_1^{(2)}} \cdots y_{n-1}^{z_{n-1}^{(2)}}), \tag{10}$$

$$(g_0 g_1^{z_1^{(3)}} \cdots g_{n-1}^{z_{n-1}^{(3)}})^{w_3} = (y_0 y_1^{z_1^{(3)}} \cdots y_{n-1}^{z_{n-1}^{(3)}}), \tag{11}$$

where $w_i = \frac{s_1^{(i)} - s_2^{(i)}}{c_2^{(i)} - c_1^{(i)}} \mod q$, $(v_i, c_j^{(i)}, s_j^{(i)})$ $(j = 1, 2)$ is the output of the interaction between $\mathcal{P}$ and $\mathcal{V}$, $z_l^{(i)}$ is the random oracle reply for $H(y_l, g_1, \ldots, g_{n-1}, y_1, \ldots, y_{n-1})$, for $l \in \{1, \ldots, n-1\}$ and $i = 1, 2, 3$ respectively.

For $i = 2, 3$, we want to ensure that for a particular $l^* \in \{1, \ldots, n-1\}$, $z_{l^*}^{(1)} \neq z_{l^*}^{(i)}$ if g_{l^*} is where the DL problem implanted, and $z_l^{(1)} = z_l^{(i)}$ for $l \in \{1, \ldots, n-1\} \backslash \{l^*\}$. Recall that we define $z_l = H(y_l, y_1, \ldots, y_{n-1})$. Using the technique in [3], we can define $H(*, y_1, \ldots, y_{n-1})$ where $* \in \{1, \ldots, n-1\}$ in one shot, when $\mathcal{A}$ queries the random oracle for $H(*, y_1, \ldots, y_{n-1})$ the first time.

From $\frac{(9)}{(10)}$ and $\frac{(9)}{(11)}$, we can obtain

$$y_1^{(z_{l^*}^{(1)} - z_{l^*}^{(2)})} = g_{l^*}^{(z_{l^*}^{(1)} w_1 - z_{l^*}^{(2)} w_2)} g_0^{(w_1 - w_2)}, \tag{12}$$

$$y_1^{(z_{l^*}^{(1)} - z_{l^*}^{(3)})} = g_{l^*}^{(z_{l^*}^{(1)} w_1 - z_{l^*}^{(3)} w_3)} g_0^{(w_1 - w_3)}. \tag{13}$$

Let $Z_1 = z_{l^*}^{(1)}$, $Z_2 = z_{l^*}^{(2)}$, $Z_3 = z_{l^*}^{(3)}$. According to the multiple-forking algorithm, $Z_1 \neq Z_2 \mod q$ and $Z_1 \neq Z_3 \mod q$. From (12) and (13),

$$g_{l^*}^{(\frac{Z_1 w_1 - Z_2 w_2}{Z_1 - Z_2} - \frac{Z_1 w_1 - Z_3 w_3}{Z_1 - Z_3})} = g_0^{(\frac{w_1 - w_3}{Z_1 - Z_3} - \frac{w_1 - w_2}{Z_1 - Z_2})}. \tag{14}$$

The solution of the DL problem instance is

$$\log_{g_{l^*}} g_0 = \left(\frac{Z_1 w_1 - Z_2 w_2}{Z_1 - Z_2} - \frac{Z_1 w_1 - Z_3 w_3}{Z_1 - Z_3} \right) \Bigg/ \left(\frac{w_1 - w_3}{Z_1 - Z_3} - \frac{w_1 - w_2}{Z_1 - Z_2} \right) \mod q.$$

The rest of the analysis is similar to that for the 2-element case, and is not repeated.

3.5 Efficiency

We exploit the fact that an n-element multi-exponentiation (a computation of the form $a_1^{x_1} \cdots a_n^{x_n}$) can be performed significantly more efficiently than n exponentiations by the windows methods [2, 12, 20, 21]. A multi-exponentiation requires a number of multiplications and squarings. For simplicity, we assume multiplication and squaring have the same complexity, and we use t_{m} to denote the computational cost of one modular multiplication in group $\mathbb{G}$. Let $\ell = |\mathbb{Z}_q|$. Setting $\ell = 160$ and the window size w to be 1, the average costs of one exponentiation, 2-element multi-exponentiation and 4-element multi-exponentiation in $\mathbb{G}$ are about $t_{1\mathrm{xp}} = 1.488\ell t_{\mathrm{m}}$, $t_{2\mathrm{xp}} = 1.739\ell t_{\mathrm{m}}$, and $t_{4\mathrm{xp}} = 1.863\ell t_{\mathrm{m}}$ respectively. For n-element multi-exponentiation, its cost increases with n and seems to be converging to $t_{n\mathrm{xp}} = 1.988\ell t_{\mathrm{m}}$ when n reaches 15. We obtain these figures from [2, Theorem 3.4].

An efficiency comparison between our protocol and the CP protocol is given in Table 1. For the equality of 2 discrete logarithms, our protocol offers about 40% computational cost saving, and saves one group element in terms of the communication overhead. The savings are more for $n > 2$, which basically reduce the computation and the communication requirements from $O(n)$ to $O(1)$. However, we remark

Table 1 Comparisons between our protocol and Chaum-Pedersen protocol

Protocol	Chaum-Pedersen [7]		Our protocol	
Number of DLs	2	n	2	n
Prover's operation	$2t_{1\text{xp}} = 2.976\ell t_{\text{m}}$	$n \cdot 1.488\ell t_{\text{m}}$	$t_{2\text{xp}} = 1.739\ell t_{\text{m}}$	$t_{n\text{xp}} = 1.988\ell t_{\text{m}}$
Verifier's operation	$2t_{2\text{xp}} = 3.478\ell t_{\text{m}}$	$n \cdot 1.739\ell t_{\text{m}}$	$t_{4\text{xp}} = 1.863\ell t_{\text{m}}$	$t_{n\text{xp}} = 1.988\ell t_{\text{m}}$
Communication	$2\lvert\mathbb{G}\rvert + 2\ell$	$n\lvert\mathbb{G}\rvert + 2\ell$	$1\lvert\mathbb{G}\rvert + 2\ell$	
Model	Standard		Random Oracle	
Soundness	Unconditional		Computational	

that our soundness only holds against adversaries with limited computational power and is only guaranteed by our non-tight reduction. For situation where unconditional soundness is more important and efficiency is less concerned, we still recommend to use Chaum-Pedersen's protocol instead.

4 Applications in Efficient Signature Schemes

The well-known CP protocol has been widely used as a building block in many cryptosystems. For examples, applying the Fiat-Shamir transformation technique [13] on the CP protocol, several signature schemes have been constructed [8, 15]. We use our protocol to construct two efficient signature schemes.

4.1 CDH-Based Signature Scheme

The high-level idea behind our first signature scheme is as follows. For a private/public key pair $(x, y_1 = g^x)$, the signature on message m consists of h and $y_2 = h^x$, where h is a group element computed from a hash function taking m as part of the input. This involves checking the equality of discrete logarithm. Furthermore, h is computed using the "random selector bit" idea proposed in [17], in which $h_b = H_0(m, b)$ where b is a random bit chosen by the signer. In the simulation of the random oracle H_0, the simulator knows the discrete logarithm of h_b but does not know that of h_{1-b} for a random b. The former case enables the signing oracle to be able to simulate the signing queries while the latter case ensures that the simulator can solve the underlying CDH problem.

4.1.1 Construction

Let $\mathbb{G}$ be a group with ℓ_q-bit prime order q where ℓ_q is the security parameter. Let g be a random generator in $\mathbb{G}$. Let $H_0 : \{0,1\}^* \to \mathbb{G}$, $H_1 : \mathbb{G}^2 \to \mathbb{Z}_q^*$ and $H_2 : \mathbb{G}^3 \to \mathbb{Z}_q^*$ be three cryptographic functions. All of the above constitute the global system parameters. Our signature scheme S1 is defined as follows:

KeyGen: Pick $x \in_R \mathbb{Z}_q^*$ as the private key and compute $y = g^x$ as the public key.

Sign: With the private key $x \in \mathbb{Z}_q^*$ and a message $m \in \{0,1\}^*$ as inputs, output the previously generated signature if m has been signed before. Otherwise, sign as below:

1. pick a random number k from $\mathbb{Z}_q^*$ and a random bit $b \in \{0, 1\}$;
2. compute $h = H_0(m, b)$ and $u = h^x$;
3. compute $z = H_1(y, u)$ and $v = (g^z h)^k$;
4. compute $e = H_2(y, u, v)$ and $s = k - xe \mod q$.

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

Verify: To verify $\sigma = (e, u, s, b)$ on a message m, compute $h = H_0(m, b)$, $z = H_1(y, u)$ and $v' = (g^z h)^s (y^z u)^e$; output 1 if $e = H_2(y, u, v')$ holds, 0 otherwise.

To see the consistency,

$$v' = (g^z h)^s (y^z u)^e = (g^z h)^s (g^z h)^{xe} = (g^z h)^{s+xe} = (g^z h)^k = v.$$

4.1.2 Stateless Variant

In the above signature scheme, the signer is stateful as she must maintain a record for all the previously issued signature/message pairs, which is the major disadvantage. Fortunately, it can be easily resolved by a standard technique which enforces the random choices to be uniquely determined by the message m. As a result, the same signature will be given whenever the same message is signed. As suggested in [15, 17], the signer could store an additional secret seed r as part of its private key and compute $(b, k) = PRF_r(m)$ where PRF is a pseudorandom function.

4.1.3 Security

The security of our scheme S1 can be ensured by the following theorem.

Theorem 2 *Our signature scheme* S1 *is* EUF-CMA *secure in the random oracle model, under the CDH assumption in group* $\mathbb{G}$. *Concretely, if there exists a* t*-time adversary* $\mathcal{F}$ *that has a non-negligible advantage* ε *against the* EUF-CMA *security of our scheme* S1, *making at most* q_S *signing queries and* q_{H_i} *queries on hash functions* H_i *with* $i = 0, 1, 2$, *then there exists an algorithm* $\mathcal{B}$ *that can solve the CDH problem with advantage*

$$\varepsilon' \geqslant \frac{\varepsilon}{2} - \frac{\eta}{2} - \frac{1 + q_{H_0} + q_{H_1} + q_s}{2^{\ell_q + 1}} \quad \textit{within time } t' \leq t + (q_s + q_{H_0})(3t_{1\text{xp}} + 1t_{4\text{xp}}),$$

where η *denotes the success probability to break the soundness of our protocol in Fig. 2, and* $t_{1\text{xp}}$ *denotes the computational cost of an exponentiation in* $\mathbb{G}$, *and* $t_{4\text{xp}}$ *denotes the computational cost of a* 4*-element multi-exponentiation in* $\mathbb{G}$.

Proof We describe how to construct an algorithm $\mathcal{B}$ which can output g^{ab} when given a CDH instance $(g, g^a, g^b) \in \mathbb{G}^3$, by interacting with $\mathcal{F}$ as a challenger in the EUF-CMA game.

Setup. $\mathcal{B}$ defines the public key as $y = g^a$, and gives y to $\mathcal{F}$. Note that the corresponding secret key is implicitly set to be a, which is unknown to $\mathcal{B}$.

Queries. In this phase, $\mathcal{B}$ answers a series of signing queries and hash queries for $\mathcal{F}$.

- **Hash queries**. To answers the hash queries for $\mathcal{F}$, $\mathcal{B}$ maintains three hash query lists H_0^{list}, H_1^{list}, H_2^{list}, and a signature list S^{list}. All these lists are initially empty. Hash function queries on H_1 or H_2 are manipulated in a natural way: $\mathcal{B}$ first checks whether this query is already in lists H_1^{list} or H_2^{list}. If yes, 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.

  When $\mathcal{F}$ issues a hash query $H_0(m, b_m)$, $\mathcal{B}$ first checks whether (m, b_m) is already in the list H_0^{list}. If yes, the previously assigned value is returned. Otherwise, $\mathcal{B}$ first follows the *signature generation procedure* described as below:

  1. select three random numbers (k, r, s) from $\mathbb{Z}_q^*$ and a random bit b'_m from $\{0, 1\}$;
  2. compute $u = y^k$;
  3. set $h = H_0(m, b'_m) = g^k$ and insert the tuple (m, b'_m, h, k) into list H_0^{list};
  4. query the hash function H_1 to obtain $H_1(y, u) = z$;
  5. compute $v = (g^z h)^s (y^z u)^r$;
  6. if $H_2(y, u, v)$ is already defined, $\mathcal{B}$ outputs "fail" and `aborts` (denote this event by E_1); else, $\mathcal{B}$ sets $H_2(y, u, v) = r$ and inserts the tuple (y, u, v, r) into list H_2^{list};
  7. $\mathcal{B}$ defines the signature on message m to be $\sigma = (u, r, s, b_m)$ and inserts the tuple (m, σ) into the list S^{list};
  8. $\mathcal{B}$ selects a random number ρ from $\mathbb{Z}_q^*$ and computes $(g^b)^\rho$ and inserts the tuple $(m, 1 - b'_m, (g^b)^\rho, \rho)$ into list H_0^{list};

  Finally, if $b_m = b'_m$, $\mathcal{B}$ returns h as the answer. Otherwise, it returns $(g^b)^\rho$.
- **Signing queries**. When $\mathcal{F}$ issues a signature query on message m, $\mathcal{B}$ first checks whether m is already in the list S^{list}. If yes, $\mathcal{B}$ returns the previously generated signature to $\mathcal{F}$. Otherwise, $\mathcal{B}$ follows the above *signature generation procedure* and returns the resulted signature to $\mathcal{F}$.

Forge. Finally, $\mathcal{F}$ outputs a message/signature pair $(\tilde{m}, \tilde{\sigma})$, where $\tilde{\sigma^*} = (\tilde{u}, \tilde{r}, \tilde{s}, \tilde{b})$. $\mathcal{B}$ first checks whether $\mathcal{F}$ has asked the hash queries $H_0(\tilde{m}, *)$ before. If not, $\mathcal{B}$ outputs "fail" and `aborts` (denote this event by E_2). Next, $\mathcal{B}$ checks whether $\tilde{b} = b'_{\tilde{m}}$. If yes, $\mathcal{B}$ outputs "fail" and `aborts` (denote this event by E_3). Otherwise, $\mathcal{B}$ searches list H_0^{list} to find the tuple $(\tilde{m}, \tilde{b}, (g^b)^{\tilde{\rho}}, \tilde{\rho})$ which indicates that $\tilde{h} = H_0(\tilde{m}, \tilde{b}) = (g^b)^{\tilde{\rho}}$. Then, $\mathcal{B}$ forms the following equation

$$\tilde{u} = (H_0(\tilde{m}, \tilde{b}))^a = ((g^b)^{\tilde{\rho}})^a = (g^{ab})^{\tilde{\rho}},$$

and then can extract the solution to the given CDH instance as $\tilde{u}^{\frac{1}{\tilde{\rho}}}$, if $\log_g y = \log_{\tilde{h}} \tilde{u}$ holds. Otherwise, $\mathcal{B}$ outputs "fail" and `aborts`, and we denote this event by E_4.

This completes the description of the simulation. Now, we turn to analyze the events E_1, E_2, E_3 and E_4. It can easily be seen that $\Pr[E_1] \leqslant \frac{q_{H_0}+q_{H_2}+q_s}{2^{\ell_q}}$. For event E_2, since H_0 is modeled as a random oracle, we can see that $\Pr[E_2] = \frac{1}{2^{\ell_q}}$. As to event E_3, since $\mathcal{F}$ did not previously issues a signature query on message $\tilde{m}$, the bit $b'_{\tilde{m}}$ is independent of $\mathcal{F}$'s view, and hence we have $\Pr[E_3] = \frac{1}{2}$. If event E_4 happens, it immediately means that $\mathcal{F}$ can successfully attack the soundness of our protocol

described in Fig. 2. So we have $\Pr[E_4] \leqslant \eta$. Combining these results, we can easily see that $\mathcal{B}$'s success probability ε' satisfies

$$\varepsilon' \geqslant \frac{\varepsilon}{2} - \frac{\eta}{2} - \frac{1 + q_{H_0} + q_{H_2} + q_s}{2^{\ell_q+1}}.$$

From the description of the simulation, we have $\mathcal{B}$'s running time bounded by $t' \leq t + (q_s + q_{H_0})(3t_{1\text{xp}} + 1t_{4\text{xp}})$. This completes the proof of Theorem 2. □

4.2 DDH-Based Signature Scheme

The second scheme is based on the observation made in [15, 17]. It is unnecessary to generate a new h for each message if we made the (stronger) DDH assumption. The public key contains (g, h, y_1, y_2) and a signature is simply a proof such that the public key is a DDH tuple.

Let $\mathbb{G}$ be a group with ℓ_q-bit prime order q where ℓ_q is the security parameter. Let g be a random generator of $\mathbb{G}$, and let $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_q^*$ be two cryptographic hash functions (which are modeled by random oracles in the security proof). Our DDH-based signature scheme S2 is specified as below:

KeyGen Pick $x \in_R \mathbb{Z}_q^*$, $h \in_R \mathbb{G}$, output x as the private key and $(h, y_1 = g^x, y_2 = h^x)$ as the public key.

Sign Given the private key $x \in \mathbb{Z}_q^*$, the corresponding public key (y_1, y_2), and a message $m \in \{0,1\}^*$, the signer generates a signature as below:

1. compute $z = H_1(y_1, y_2)$; (pre-computable from the public key)
2. compute $u = g^z h$; (pre-computable from the public key too)
3. pick $k \in_R \mathbb{Z}_q^*$, and compute $v = u^k = (g^z h)^k$; (independent of the message)
4. output $(e = H_2(m, y_1, y_2, v), s = k - xe \mod q)$.

Sign is derived from our protocol using the Fiat-Shamir heuristic: $v = (g^z h)^k$ is the commitment, e is the challenge, and $s = k - xe$ is the response.

Verify Given the public key (y_1, y_2), and a signature (e, s) on message m, compute $z = H_1(y_1, y_2)$, and $v' = (g^z h)^s (y_1^z y_2)^e$; return 1 if $e = H_2(m, y_1, y_2, v')$, 0 otherwise.

For a valid signature (e, s) on m, $v' = (g^z h)^s (y_1^z y_2)^e = (g^z)^{s+xe} h^{s+xe} = (g^z h)^k = v$.

4.2.1 Pre-computation and Offline/Online Signing

For a given signer, observe that the values $z = H_1(y_1, y_2), u = g^z h$ and $w = y_1^z y_2$ are invariants for every signature verification, and hence they can be pre-computed. This makes Sign only takes one online exponentiation since $u = g^z h$ can be pre-computed without the private key x too. This also saves Verify from one 4-element multi-exponentiation to one 2-element multi-exponentiation, and is helpful in the situation where the verifier receives signatures from the same signer more than once.

A signing algorithm can be split into offline and online stages. Offline computation refers to computation that may be performed before the message to be signed

is known, which possibly includes pre-selecting random numbers for a probabilistic signature scheme. In contrast, online computation must be done after the message is known. Our Sign algorithm can be naturally split. The offline stage generates the offline token (k, v). The online stage outputs the signature (e, s) by one hash function evaluation and one modular multiplication, which is nearly the least one may expect for a hash-based signature.

4.2.2 Security

The security of our scheme S2 can be ensured by the following theorem.

Theorem 3 *Our signature scheme* S2 *is* EUF-CMA *secure in the random oracle model, under the DDH assumption in group* $\mathbb{G}$. *Concretely, if there exists a* t*-time adversary* $\mathcal{F}$ *that has a non-negligible advantage* ε *against the* EUF-CMA *security of our scheme* S2, *making at most* q_S *signing queries,* q_{H_1} *queries on hash function* H_1, *and at most* q_{H_2} *queries on hash function* H_2, *then there exists an algorithm* $\mathcal{B}$ *that can solve the DDH problem with*

$$\text{advantage } \varepsilon' \geqslant \varepsilon - \eta - \frac{\eta + q_{H_2} + q_s}{2^{\ell_q}} \quad \text{within time } t' \leq t + (q_s + 1)t_{2\text{xp}},$$

where η *denotes the success probability to break the soundness of our protocol in Fig.* 2, *and* $t_{2\text{xp}}$ *denotes the computational cost of a* 2*-element multi-exponentiation in* $\mathbb{G}$.

Proof We describe how to construct an algorithm $\mathcal{B}$ which can determine if (g, h, y_1, y_2) is a DDH tuple, by interacting with $\mathcal{F}$ as a challenger in the EUF-CMA game.

Setup. $\mathcal{B}$ gives g and (h, y_1, y_2) as the public key to $\mathcal{F}$.

Queries. In this phase, $\mathcal{B}$ answers a series of signing queries and hash queries for $\mathcal{F}$. $\mathcal{B}$ maintains two hash query lists H_1^{list} and H_2^{list}, as well as a signature list S^{list}, which are initially empty. $\mathcal{B}$ answers these queries for $\mathcal{F}$ as below:

- **Hash queries**. When a query $H_1(y_1, y_2)$ (or $H_2(m, y_1, y_2, v)$) is issued, $\mathcal{B}$ first determines whether it has been asked before by searching list H_1^{list} (or H_2^{list}). If yes, $\mathcal{B}$ returns the previously defined value. Otherwise, $\mathcal{B}$ responds with a random value from the appropriate domain as the answer, and updates the corresponding list.
- **Signing queries**. When $\mathcal{F}$ issues a signature query on message m, $\mathcal{B}$ proceeds as follows:
  1. randomly choose two numbers e, s uniformly from $\mathbb{Z}_q^*$;
  2. query hash function H_1 to obtain $H_1(y_1, y_2) = z$;
  3. compute $v = (g^z h)^s (y_1^z y_2)^e$;
  4. if $H_2(m, y_1, y_2, v)$ is already defined, $\mathcal{B}$ outputs a random bit and `aborts` (denote this event by E_1); else, define $H_2(m, y_1, y_2, v) = e$ and return $\sigma = (e, s)$ as the signature on message m.

Forge. Finally, $\mathcal{F}$ outputs a forged signature $(\tilde{\sigma}, \tilde{s})$ on a message $\tilde{m}$ which has never been issued to the signature queries. $\mathcal{B}$ checks whether $(\tilde{\sigma}, \tilde{s})$ is a valid signature on message $\tilde{m}$. If yes, $\mathcal{B}$ outputs 1 to indicate that the challenge tuple is indeed a "DDH tuple". Otherwise, $\mathcal{B}$ outputs 0.

This completes the description of the simulation. Now, we proceed to analyze the probability that $\mathcal{B}$ outputs 1.

If (g, h, y_1, y_2) is a "DDH tuple" and event E_1 does not happen, then we can see that the simulation provided for $\mathcal{F}$ is perfect. Since e and s are randomly chosen from $\mathbb{Z}_q^*$, v is a random element in $\mathbb{G}$, and hence the probability that $H_2(m, y_1, y_2, v)$ is already defined is at most $\frac{q_s+q_{H_2}}{2^{\ell_q}}$. Therefore, it can be easily seen that, if (g, h, y_1, y_2) is a "DDH tuple", $\mathcal{B}$ will output 1 with probability at least $\varepsilon - \frac{q_s(q_s+q_{H_2})}{2^{\ell_q}}$.

On the other hand, if y_2 is "random", with probability $1 - \frac{1}{2^{\ell_q}}$, it is not a "DDH tuple", i.e., $\log_g y_1 \neq \log_h y_2$. Then the valid signature $(\tilde{e}, \tilde{s})$ immediately implies that $\mathcal{F}$ can successfully attack the soundness of our protocol described in Fig. 2. Since such probability is bounded by η, we can see that, in this case, $\mathcal{B}$ outputs 1 with probability at most $(1 - \frac{1}{2^{\ell_q}})\eta$.

Putting all together and according to Definition 3, we have that $\mathcal{B}$'s advantage in attacking the DDH problem in group $\mathbb{G}$ is at least $\varepsilon' \geq \varepsilon - \eta - \frac{\eta+q_s(q_s+q_G)}{2^{\ell_q}}$. From the description of the simulation, we can see that $\mathcal{B}$'s running time is bounded by $t' \leq t + (q_s + 1)t_{2\text{xp}}$. This concludes the proof of Theorem 3. □

4.3 Efficiency Comparisons

Table 2 presents a detailed comparison between our schemes with existing related signature schemes. To date, there are three signature schemes with tight security reductions related to the CDH assumption: the EDL scheme in [7, 16] which is proven secure in [14], the improved scheme by Katz and Wang [17] (denoted by KW-CDH), and the scheme by Chevallier-Mames [8] (denoted by C-M), which is the most efficient among these three schemes. Compared with C-M, our CDH scheme offers

Table 2 Efficiency comparisons between different signature schemes

Scheme	Sign	Verify	Signature size	Public key size
CDH based				
EDL [14]	$3t_{1\text{xp}} = 4.464\ell t_{\text{m}}$	$2t_{4\text{xp}} = 3.726\ell t_{\text{m}}$	$\lvert\mathbb{G}\rvert + 2\lvert\mathbb{Z}_q\rvert + (\ell + 31)$	$1\lvert\mathbb{G}\rvert$
KW-CDH [17]	$3t_{1\text{xp}} = 4.464\ell t_{\text{m}}$	$2t_{4\text{xp}} = 3.726\ell t_{\text{m}}$	$\lvert\mathbb{G}\rvert + 2\lvert\mathbb{Z}_q\rvert + 1$	$1\lvert\mathbb{G}\rvert$
C-M [8]	$3t_{1\text{xp}} = 4.464\ell t_{\text{m}}$	$2t_{4\text{xp}} = 3.726\ell t_{\text{m}}$	$\lvert\mathbb{G}\rvert + 2\lvert\mathbb{Z}_q\rvert$	$1\lvert\mathbb{G}\rvert$
Our Scheme S1	$t_{1\text{xp}} + t_{2\text{xp}} = 3.227\ell t_{\text{m}}$	$t_{4\text{xp}} = 1.863\ell t_{\text{m}}$	$\lvert\mathbb{G}\rvert + 2\lvert\mathbb{Z}_q\rvert + 1$	$1\lvert\mathbb{G}\rvert$
DDH-based				
KW-DDH [17]	$2t_{1\text{xp}} = 2.976\ell t_{\text{m}}$	$2t_{2\text{xp}} = 3.478\ell t_{\text{m}}$	$2\lvert\mathbb{Z}_q\rvert$	$3\lvert\mathbb{G}\rvert$
Our Scheme S2	$t_{1\text{xp}} = 1.488\ell t_{\text{m}}$	$t_{2\text{xp}} = 1.739\ell t_{\text{m}}$	$2\lvert\mathbb{Z}_q\rvert$	$3\lvert\mathbb{G}\rvert$

50% computational cost saving in verification. In [17], Katz and Wang presented a signature scheme (denoted by KW-DDH) with tight security reduction to the DDH assumption. Both the computational cost for signing and verification of our DDH-based scheme are reduced to 50%, where the comparison is made by instantiating both DDH-based schemes using the same security parameter. Even without signer-specific pre-computation by the verifier, verification only takes $1.863\ell t_{\mathrm{m}}$ which remains competitive. We remark that the aim of [17] is to achieve a tight security reduction. To compensate for the tightness, we need to use a larger security parameter, which results in a larger signature size. However, the 50% saving in computational cost means that our scheme can afford a larger security parameter without incurring too much extra computation time. On the other hand, to the best of our knowledge, we are not aware of any example of concrete attacks on a signature scheme which corresponds to the lost in the security reduction.

5 Conclusion

We investigated the protocol for the simultaneous discrete logarithms problem—a fundamental building block appears in many cryptosystems of different flavors. We noted that such a protocol is usually used in a non-interactive form by applying the Fiat-Shamir heuristics which inevitably relies on the random oracle model. The adversary is also assumed to be computationally bounded in these applications. We thus proposed an efficient zero-knowledge argument system in the random oracle model which gives a better performance in terms of both the computational cost and the communication size. We believe that our protocol can also be used to improve other cryptographic cryptosystems. In particular, we proposed a signature scheme that only takes one (offline) exponentiation to sign, without pairing, based on the standard decisional Diffie-Hellman assumption.

We remark that our technique can be generalized for proving linear relations other than the simple equality. It may be interesting to see if other zero-knowledge argument systems can be speeded up using a similar technique.

Acknowledgements Part of this work was done while the first author was a Ph.D. student in Courant Institute of Mathematical Sciences, New York University, and while the second and the third authors were postdoctoral fellows in School of Information Systems, Singapore Management University. We would like to express our gratitude to the anonymous reviewers for their helpful comments which improve a few aspects of our paper. Thanks also go to Dong Zheng who helped in the initial stage of this research. This work is partially supported by the Office of Research, Singapore Management University. It is also supported by the National Science Foundation of China under Grant Nos. 60903178, 61070217, 61005049 and 61133014, the Fundamental Research Funds for the Central Universities under Grant No. 21610204, and the Guangdong Provincial Science and Technology Project under Grand No. 2010A032000002.

References

1. Ateniese, G.: Verifiable encryption of digital signatures and applications. ACM Trans. Inform. Syst. Secur. (TISSEC) **7**(1), 1–20 (2004)
2. Avanzi, R.M.: On multi-exponentiation in cryptography. Cryptology ePrint Archive, Report 2002/154 (2002)

3. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30–November 3, 2006, pp. 390–399. ACM, New York (2006)
4. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security, CCS 1993, Fairfax, Virginia, USA, 3–5 November 1993, pp. 62–73 (1993)
5. Boldyreva, A., Palacio, A., Warinschi, B.: Secure proxy signature schemes for delegation of signing rights. J. Crypt. (2011). doi:10.1007/s00145-010-9082-x
6. Camenisch, J., Maurer, U.M., Stadler, M.: Digital payment systems with passive anonymity-revoking trustees. J. Comput. Secur. **5**(1), 69–90 (1997)
7. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) Advances in Cryptology—CRYPTO'92, 12th Annual International Cryptology Conference Proceedings, Santa Barbara, California, USA, 16–20 August 1992. Lecture Notes in Computer Science, vol. 740, pp. 89–105. Springer, Berlin (1992)
8. Chevallier-Mames, B.: An efficient CDH-based signature scheme with a tight security reduction. In: Shoup, V. (ed.) Advances in Cryptology—CRYPTO 2005: 25th Annual International Cryptology Conference Proceedings, Santa Barbara, California, USA, 14–18 August 2005. Lecture Notes in Computer Science, vol. 3621, pp. 511–526. Springer, Berlin (2005)
9. Chow, S.S.M., Liu, J.K., Wong, D.S.: Robust receipt-free election system with ballot secrecy and verifiability. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10–13 February 2008, pp. 81–94. The Internet Society, Reston (2008)
10. Chow, S.S.M., Ma, C., Weng, J.: Zero-knowledge argument for simultaneous discrete logarithms. In: Thai, M.T., Sahni, S. (eds.) Computing and Combinatorics, 16th Annual International Conference, Proceedings COCOON 2010, Nha Trang, Vietnam, 19–21 July 2010. Lecture Notes in Computer Science, vol. 6196, pp. 520–529. Springer, Berlin (2010)
11. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) Advances in Cryptology—EUROCRYPT'97, International Conference on the Theory and Application of Cryptographic Techniques, Proceeding, Konstanz, Germany, 11–15 May 1997. Lecture Notes in Computer Science, vol. 1233, pp. 103–118. Springer, Berlin (1997)
12. Dimitrov, V.S., Jullien, G.A., Miller, W.C.: Complexity and fast algorithms for multiexponentiations. IEEE Trans. Comput. **49**(2), 141–147 (2000)
13. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology—CRYPTO'86, Proceedings, Santa Barbara, California, USA, 1986. Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer, Berlin (1986)
14. Goh, E.-J., Jarecki, S.: A signature scheme as secure as the Diffie-Hellman problem. In: Biham, E. (ed.) Advances in Cryptology—EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Warsaw, Poland, 4–8 May 2003. Lecture Notes in Computer Science, vol. 2656, pp. 401–415. Springer, Berlin (2003)
15. Goh, E.-J., Jarecki, S., Katz, J., Wang, N.: Efficient signature schemes with tight reductions to the Diffie-Hellman problems. J. Cryptol. **20**(4), 493–514 (2007). Journal version of [17] and [14]
16. Jakobsson, M., Schnorr, C.-P.: Efficient oblivious proofs of correct exponentiation. In: Preneel, B. (ed.) Secure Information Networks: Communications and Multimedia Security, IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS'99), 20–21 September 1999, Leuven, Belgium. IFIP Conference Proceedings, vol. 152, pp. 71–86. Kluwer, Dordrecht (1999)
17. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: Jajodia, S., Atluri, V., Jaeger, T. (eds.) Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, Washington, DC, USA, 27–30 October 2003, pp. 155–164. ACM, New York (2003)
18. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym Systems. In: Heys, H.M., Adams, C.M. (eds.) Selected Areas in Cryptography, 6th Annual International Workshop, SAC'99, Proceedings, Kingston, Ontario, Canada, 9–10 August 1999. Lecture Notes in Computer Science, vol. 1758, pp. 184–199. Springer, Berlin (1999)
19. Ma, C., Weng, J., Li, Y., Deng, R.H.: Efficient discrete logarithm based multi-signature scheme in the plain public key model. Des. Codes Cryptogr. **54**(2), 121–133 (2010)
20. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (2001)

21. Möller, B.: Algorithms for multi-exponentiation. In: Vaudenay, S., Youssef, A.M. (eds.) Selected Areas in Cryptography, 8th Annual International Workshop, Revised Papers, SAC 2001, Toronto, Ontario, Canada, 16–17 August 2001. Lecture Notes in Computer Science, vol. 2259, pp. 165–180. Springer, Berlin (2001)
22. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptol. **13**(3), 361–396 (2000)
23. Shoup, V.: Practical Threshold Signatures. In: Preneel, B. (ed.) Advances in Cryptology—EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Proceeding, Bruges, Belgium, 14–18 May 2000. Lecture Notes in Computer Science, vol. 1807, pp. 207–220. Springer, Berlin (2000)
24. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. J. Cryptol. **15**(2), 75–96 (2002)
25. Stadler, M.: Publicly verifiable secret sharing. In: Maurer, U.M. (ed.) Advances in Cryptology—EUROCRYPT'96, International Conference on the Theory and Application of Cryptographic Techniques, Proceeding, Saragossa, Spain, 12–16 May 1996. Lecture Notes in Computer Science, vol. 1070, pp. 190–199. Springer, Berlin (1996)