How to Generate Universally Verifiable Signatures in Ad-Hoc Networks *

KyungKeun Lee† JoongHyo Oh‡ SangJae Moon§

November 3, 2005

Abstract

This paper addresses the problem of making signatures of one domain (an ad-hoc network) available in another domain (the Internet). Universal verifiability is a highly desirable property when signed documents need to be permanently non-repudiable so as to prevent dishonest signers from disavowing signatures they have produced. As a practical solution, we construct a new signature scheme where a valid signature should be generated by a couple of distinct signing keys. In the random oracle model, the signature scheme is provably secure in the sense of existential unforgeability under adaptive chosen message attacks assuming the hardness of the computational Diffie-Hellman problem in the Gap Diffie-Hellman groups.

Keywords: Ad-hoc networks, Universal verifiability, Interoperability, Digital signature, Diffie-Hellman problem, Bilinear map, Random oracle model

Version Info. This paper is a full version of [LOM05] written by the same authors. Most notably, this full paper presents a rigorous, formal proof of Lemma 3 and 5 in the appendix A. As a consequence, the security of the proposed scheme is *tightly* related to the computational Diffie-Hellman problem. This implies that the security reduction of our resultant scheme is quite comparable to those of the recent results in [GJ03, CM05].

*This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment).

†The School of EECS, Kyungpook National University, Korea, `skywalker@mail.knu.ac.kr`

‡The Korea Financial Telecommunications and Clearings Institute, Korea, `jhoh@kftc.or.kr`

§The School of EECS, Kyungpook National University, Korea, `sjmoon@knu.ac.kr`

Contents

1 Introduction

In modern cryptography, digital signatures are among the most fundamental and prominent tools for realizing security in open distributed systems and networks. In a conventional Public Key Infrastructure (PKI) environment [ITU97], it is easy to confirm the authorship of a signed document by verifying the signature with a certificate of a given public key and, simultaneously, by checking whether the corresponding certificate is revoked, communicating with on-line Certificate Authority (CA). Within an *ad-hoc* environment, a user can also generate and verify digital signatures with the assistance of pre-established ad-hoc CA [ZH99].

Nevertheless, signatures generated from an ad-hoc network are neither easily validated nor directly compatible in any infra network since the ad-hoc network is, in general, assumed to be physically isolated from the infra network, i.e, *autonomous*. For instance, suppose that Alice is an (ad-hoc) signer and Bob is a (mobile) verifier. When Bob receives Alice's signature, he is able to verify her signature and to identify her interacting with an on-line ad-hoc CA. Now, Bob switches to an infra network such as the Internet disconnecting the ad-hoc network channel. Even if this signature was verified to be genuine before leaving the ad-hoc network, the signature is not verifiable to any other party who has received the signature from Bob within the infra network because her certificate is inapplicable outside its domain. Despite Bob's efforts to retrieve Alice's certificate status information, he will not succeed due to difficulties in communication with the ad-hoc CA. Even if the current status of her certificate can be obtained, the certificate will still be of dubious standing and even extraneous since the ad-hoc CA is definitely not a trusted authority in the new domain. Alice is then exempt from responsibility of the signature she produced. Thus, it is highly desirable to make signatures non-repudiable in order that signers should be permanently undeniable even when they are in ad-hoc networks. A trivial solution for the long-term non-repudiation is to make the signer participate in so-called a posteriori arbitration when a dispute occurs. Intuitively this requires a strong assumption that any suspicious signer would willingly take part in the arbitration; unfortunately, this assumption is not realistic.

The alternative solution proposed in this paper is to provide the *universal verifiability* for signatures. Our scheme is an a priori and non-interactive (for signers) protocol in which a mobile verifier is given an ad-hoc signature as well as a corresponding infra (actually encrypted) signature. At the same time, the verifier also obtains an additional zero-knowledge argument that demonstrates the signer's possession of (ad-hoc and infra) signing keys used for generating signatures. Interacting with a trusted "notary," the verifier will finally obtain the translation of the given ad-hoc signature while authenticating the signer's identity. This translation becomes identical to the infra signature as desired.

We construct a universally verifiable signature scheme (refer to the P2DL signature) by extending the verifiably encrypted signature scheme proposed by Boneh *et al.* [BGLS03]. We also extend the zero-knowledge proof of discrete-log equality [CP93] into a zero-knowledge proof of two discrete logarithms possession and give its formal analysis. The signature scheme presented here offers provable security against adaptive chosen message attacks under the Gap Diffie-Hellman assumption [OP01] in the random oracle model [BR93].

1.1 Motivation

Most signature schemes provide so-called "public verifiability," as long as the validation of given a message-signature pair with some redundant parameters is believed to be true. In accordance with this validation, the authenticity of corresponding signer's public (verification) key is also requisite after retrieving the signer's certificate status information from the CA according to X.509 [ITU97]. As such, the verification of signatures are on two points: the validation of given the signature, message, and parameters, plus the signer's public-key validity via revocation check. (In short, *signature validation* and *revocation*.)

In this paper, we pay attention to the verifiability of signature in two-domain environment, especially in an ad-hoc network and the Internet. Ad-hoc networks are substantially autonomous. Thus, there is no communication channel from an ad-hoc network to the Internet (and vice versa), whatsoever wired or wireless channel, by definition of the ad-hoc network. *If an ordinary signature is generated from an ad-hoc network, is that signature available in the Internet?* Owing to the isolation between two networks, it looks insurmountable though.

Suppose that Alice is a signer with two signing keys $SK_{\text{ad-hoc}}$ and SK_{infra} for the purpose of signature generations in ad-hoc and infra networks, respectively. She is able to generate a signature σ under her signing key $SK_{\text{ad-hoc}}$ for the ad-hoc network and to make any user verifiable with this signature within its domain. It is always verifiable for every ad-hoc user as desired. *What if this ad-hoc signature is transferred to any other domain such as the Internet?* Even if this signature is genuine via *signature validation*, it is impossible to check whether the signer's key is revoked or not. One can ask why not use an infra signing key SK_{infra} within every ad-hoc network for the public verifiability in the Internet. An analogous problem arises for ad-hoc verifiers because of the difficulties in checking the revocation of the signer's infra network public-key. Due to the limitation of ordinary signatures in the literature, we construct a new signature scheme that offers the *universal verifiability* across two isolated domains.

Definition 1 (Universal Verifiability of Two-Domain Setting)**.** A signature Σ transferred from one domain is said to be universally verifiable when a given signature is publicly verifiable in both one domain $\mathcal{D}_1$ and in the other domain $\mathcal{D}_2$.

- (Unilaterally universal verifiability). A signature generated from $\mathcal{D}_1$ is also publicly verifiable in $\mathcal{D}_2$, but a signature generated from $\mathcal{D}_2$ is not verifiable in $\mathcal{D}_1$. In this case, the signature scheme is universally verifiable in a unilateral way from $\mathcal{D}_1$ to $\mathcal{D}_2$.
- (Bilaterally universal verifiability). A signature originated from any domain, $\mathcal{D}_1$ or $\mathcal{D}_2$, is publicly verifiable in both domains $\mathcal{D}_1$ and $\mathcal{D}_2$.

1.2 Our Contributions

Supporting the unilaterally universal verifiability from an ad-hoc domain to the Internet domain, our signature scheme features several characteristics below.

- *Non-interactivity.* In the proposed scheme, signing process is not interactive. After sending a signed document to any verifier, signer does not have to attend any other protocol without listening to the verifier's feedback. It is highly desirable in most signature schemes and makes our scheme more practical.

- *Verifier's Mobility Support.* Due to the isolation between ad-hoc and infra networks, the only way to transfer signatures generated across domains is to allow verifiers mobile from an ad-hoc network to the Internet. We can imagine verifiers are mobile devices such as laptop computers, PDAs, or any other hand-held electrical conveniences.

- *Granularity.* In our scheme, a signer simply generates ordinary short signatures for use in each ad-hoc and infra networks. Next the signer computes the non-interactive zero knowledge (NIZK) in order to prove the signer's possession of two signing secret keys. Intuitively, the signature Σ comprises an ad-hoc signature σ_1, infra signature σ_2, and the redundancy for NIZK. Hence, after finishing appropriate validation steps, our signature Σ is no longer necessary. For ad-hoc verifiers, σ_1 is sufficient as a signature of the given message. On the other hand, σ_2 makes infra network users convinced that the given message-signature pair is authentic. In particular, the granularity feature makes our signature scheme attractive since shorter (ordinary) signature is enough for any verifier in each separated network.

- *Notarization.* A special-purpose trusted third party (TTP) of the Internet domain is assumed in our scheme. The notary's task is mainly to notify the revocation of signer's infra public key on behalf of the on-line infra CA. In addition, notary verifies any signature transferred from ad-hoc domain and then publishes a smaller ordinary signature for the Internet use only. Before notarization, this infra signature is encrypted under the notary's public key. This disables for any mobile verifier to spread signer's infra signature in any other domain without doing in the Internet. The presence of the trusted notary is not burdensome because it looks like a slightly modified CA. This also makes our protocol quite realistic.

- *Tight Security Reduction.* Interestingly, equipped with the redundant information for NIZK proof, the security of the proposed scheme is *tightly* related to the computational Diffie-Hellman problem with formal proof, while the co-GDH signature scheme [BLS04] adopted as underlying signatures is *loosely* related to the same hard problem. In principle, the tight security reduction is infinitely preferable for strengthening the security of protocols.

1.3 Organization

The rest of this paper is organized as follows. The following section provides an overview of relevant background. Section 3 presents the new universally verifiable signature scheme. The security of the scheme is formally analyzed in Section 4. We conclude this paper in Section 5. Finally we present the rigorous proof of Lemmas in Appendix.

2 Preliminaries

2.1 Notation and Bilinear Maps

We first consider two (distinct and multiplicative) cyclic groups $\mathcal{G}_1 = \langle g_1 \rangle$ and $\mathcal{G}_2 = \langle g_2 \rangle$ of prime order q, where the discrete logarithm problem is intractable. Let ψ be a computable isomorphism from $\mathcal{G}_1$ to $\mathcal{G}_2$, with $\psi(g_1) = g_2$.

We require a bilinear map $\hat{e} : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$, where the group $\mathcal{G}_T$ is multiplicative and the orders of all the given groups are the same: $\#\mathcal{G}_1 = \#\mathcal{G}_2 = \#\mathcal{G}_T = q$. A bilinear map satisfying the following properties is said to be an *admissible* bilinear map: (a) bilinear: for all $u \in \mathcal{G}_1, v \in \mathcal{G}_2$ and $a, b \in \mathbb{Z}_q, \hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$; (b) non-degenerate: given generators g_1 and g_2, $\hat{e}(g_1, g_2) \neq 1$; (c) computable: there is an efficient algorithm to compute $\hat{e}(u, v)$ for any $u \in \mathcal{G}_1, v \in \mathcal{G}_2$. With an isomorphism $\psi(\cdot)$, these properties lead to two more simple attributes: for any $u, w \in \mathcal{G}_1$, $\hat{e}(u, \psi(w)) = \hat{e}(w, \psi(u))$; and for any $u \in \mathcal{G}_1, v_1, v_2 \in \mathcal{G}_2$, $\hat{e}(u, v_1 v_2) = \hat{e}(u, v_1) \cdot \hat{e}(u, v_2)$. Without loss of generality, we assume these maps $\psi(\cdot)$ and $\hat{e}(\cdot)$ can be computed in one time unit for the group action, and we then call the two groups $(\mathcal{G}_1, \mathcal{G}_2)$ a bilinear group pair.

The expression $z \in_R S$ indicates that an element z is chosen randomly from the finite set S according to the uniform distribution. When parsing a string w into a sequence of fragments $U_1, U_2, \ldots, U_\ell$, we use $w \xrightarrow{\mathrm{P}} U_1 \| U_2 \| \ldots \| U_\ell$. For simplicity, the $(\bmod\ q)$ marker for operations on elements in $\mathbb{Z}_q$ is omitted.

2.2 Intractability Assumptions

We define several hard problems closely related to the Diffie-Hellman problem [DH76]. Consider a single cyclic group $\mathcal{G} = \langle g \rangle$, with $q = \#\mathcal{G}$ a prime.

Computational Diffie-Hellman Problem (CDH). For $a, b \in \mathbb{Z}_q$, given g, $g^a, g^b \in \mathcal{G}$, compute $g^{ab} \in \mathcal{G}$.

Decision Diffie-Hellman Problem (DDH). For $a, b, c \in \mathbb{Z}_q$, given g, g^a, g^b, $g^c \in \mathcal{G}$, answer with;

(i) Yes, if $c = ab$. In this case, (g, g^a, g^b, g^c) is called a Diffie-Hellman quadruple.

(ii) No, otherwise.

There is strong evidence that these two standard hard problems are closely related the hardness of computing discrete logarithms. There exist polynomial-time algorithms that either reduce the DDH to the CDH or reduce the CDH to the discrete logarithm problem. The converses of these reductions, however, are thus far not known to hold. All the three problems are widely conjectured to be intractable. A variety of cryptographic protocols have been proved secure under these standard intractability assumptions. Refer to [MW99, Bon98] for further reading.

Now suppose that two distinct cyclic groups $\mathcal{G}_1 = \langle g_1 \rangle$ and $\mathcal{G}_2 = \langle g_2 \rangle$ of prime order q form a bilinear group pair. We then obtain natural generalizations of the computational and decisional Diffie-Hellman problems with this bilinear group pair $(\mathcal{G}_1, \mathcal{G}_2)$. Using the prefix "co" implies that the given two groups are different (but with the same order).

Computational Co-Diffie-Hellman Problem (co-CDH). For $a \in \mathbb{Z}_q$, given g_1, $g_1^a \in \mathcal{G}_1$ and $h \in \mathcal{G}_2$, compute $h^a \in \mathcal{G}_2$.

Decision Co-Diffie-Hellman Problem (co-DDH). For $a, b \in \mathbb{Z}_q$, given g_1, $g_1^a \in \mathcal{G}_1$ and $h, h^b \in \mathcal{G}_2$, answer with;

(i) Yes, if $a = b$. In this case, (g_1, g_1^a, h, h^b) is called a co-Diffie-Hellman quadruple.

(ii) No, otherwise.

Here, we define co-gap Diffie-Hellman (co-GDH) groups to be group pairs $\mathcal{G}_1$ and $\mathcal{G}_2$ on which co-DDH is easy but co-CDH is hard. This class of problems—gap problems—was first introduced by Okamoto and Pointcheval [OP01] and is considered to be dual to the class of the decision problems. They also discussed several cryptographic applications based on these problems such as the undeniable signature scheme introduced by Chaum and Pederson [CP93] and designated confirmer signatures. Joux and Nguyen [JN01] also pointed out that, for the special case of supersingular elliptic curves, DDH is easy while CDH is still hard.

Definition 2 (co-GDH assumption [BGLS03])**.** We say that both $\mathcal{G}_1$ and $\mathcal{G}_2$ are (t, ϵ)-co-GDH groups if they are in the class of decision groups for co-Diffie-Hellman and no PPT algorithm (t, ϵ)-breaks co-CDH in groups $\mathcal{G}_1$ and $\mathcal{G}_2$ with sufficiently large prime q.

2.3 Co-GDH Signature Scheme

The co-GDH signature scheme due to Boneh *et al.* [BLS04] is known to be one of the shortest signatures in the literature. In this scheme, there are three algorithms, *KeyGen*, *Sign*, and *Verify*, plus a full-domain hash function $H : \{0,1\}^* \rightarrow \mathcal{G}_2$, which works as a random oracle.

Key Generation. The secret key x is chosen at random $x \in_R \mathbb{Z}_q$, and the corresponding public key y is computed by $y \leftarrow g_1^x$. The public key is an element of $\mathcal{G}_1$.

Signing. To generate a signature on a message $M \in \{0,1\}^*$ under the secret key x, compute $\sigma \leftarrow h^x$, where $h = H(M)$. Thus, the message-signature pair becomes (M, σ).

Verification. The verification algorithm takes the signer's public key y, message M, and signature σ as input. It first computes $h \leftarrow H(M)$ and checks if (g_1, y, h, σ) is a valid co-Diffie-Hellman quadruple.

We can simply make use of a bilinear map $\hat{e}$ from $\mathcal{G}_1 \times \mathcal{G}_2$ to $\mathcal{G}_T$ for co-DDH testing: $\hat{e}(g_1, \sigma) \stackrel{?}{=} \hat{e}(y, h)$. This signature scheme is existentially unforgeable under adaptive chosen message attacks [GMR88] in the random oracle model. However, the security reduction is loosely related to the CDH problem.

2.4 Proving Equality of Two Discrete Logarithms

Let $\mathcal{G} = \langle g \rangle = \langle h \rangle$ be a group of prime order q with generators g, h. Denote $\mathsf{EDL}(g, h)$ as the language of pairs $(y, \sigma) \in \mathcal{G}^2$ such that $\mathtt{dlog}_g y = \mathtt{dlog}_h \sigma$, where $\mathtt{dlog}_g y$ denotes the discrete logarithm of y with respect to the base g (identically for $\mathtt{dlog}_h \sigma$). The following is a well-known zero-knowledge proof system for the proof of equality of two discrete logarithms [CP93]. This system proceeds as follows.

- Given $(y, \sigma) \in \mathsf{EDL}(g, h)$, the prover picks $k \in_R \mathbb{Z}_q$ at random, computes $u \leftarrow g^k, v \leftarrow h^k$, and then sends them to the verifier.
- The verifier chooses $c \in_R \mathbb{Z}_q$ as a public random challenge and sends it to the prover.
- Upon receipt of c, the prover responds with $s \leftarrow k + cx$.
- The verifier checks if $g^s \stackrel{?}{=} uy^c$ and $h^s \stackrel{?}{=} v\sigma^c$ hold true.

It is well known that this proof system is both complete and sound. See [CP93] for details. Notice that this interactive proof system is a public-coin zero-knowledge since a simulator given inputs g, h, y, σ can choose c, s at random in $\mathbb{Z}_q$ and compute u and v from them. A non-interactive version of this proof system has several applications such as EDL signature schemes [GJ03, KW03, CM05] that are proved to be as secure as the CDH problem and threshold cryptosystems that are provably secure against chosen ciphertext attacks under the CDH and DDH assumptions [SG98], viewing the hash functions as random oracles [BR93].

3 P2DL Signature Scheme

3.1 New Model

The participants in our protocol are classified into three groups; *Signer*, *Verifier*, and *Notary*. Each of them plays a respective role:

Signer. She wants to produce a signature on a document in an ad-hoc network. Each signer has its ad-hoc and infra certificates that are issued by the respective ad-hoc and infra CAs. Roughly speaking, her P2DL signature contains three essential ingredients:

1. An ad-hoc signature σ_1. This is only valid in the ad-hoc network.
2. An infra signature σ_2. This is only valid in the infra network. She does not want any recipient to obtain this signature until notarization. Thus, an encrypted signature is transmitted to a receiver.
3. A zero-knowledge argument is provided to prove herself possessing a couple of signing (secret) keys, which are used in generating σ_1 and σ_2 signatures.

In general, she does not have to establish a communication channel to the infra network.

Verifier. Any verifier is assumed to be potentially mobile from the ad-hoc network to the infra network. Upon receipt of a signer's signature, the verifier can immediately verify her ad-hoc signature σ_1 within an ad-hoc network. After switching to the infra network, the verifier can obtain an infra signature σ_2 by asking the trusted notary to translate P2DL signature into an infra one.

Trusted Notary. A special-purpose TTP is assumed, the trusted notary.[1] The notary operates when a verifier makes a request of notarization. The notary performs two tasks:

1. It determines whether the given P2DL signature is valid and whether the given ad-hoc and infra signatures are generated by one person. If "YES," it publishes σ_2 as the translation of σ_1. If "NO," it notifies that the given signature is invalid.
2. During notarization, the notary offers a public-key certification service upon receiving the signer's (infra) public key on behalf of the on-line (infra) CA.

[1] Notice that the notary is an entity only available in the infra network while signers stay within the ad-hoc network (when sending their signatures to verifiers). In contrast, verifiers are possibly mobile from an ad-hoc network to the infra network. Our signature scheme features the mobility nature of verifiers.

3.2 P2DL Signature Scheme

The protocol uses two hash functions. The security analysis will view these hash functions as random oracles.

$$H_1 : \{0,1\}^* \times \mathbb{Z}_q \to \mathcal{G}_2,\ \ H_2 : (\mathcal{G}_2)^9 \to \mathbb{Z}_q$$

Key generation. Each signer is assumed to hold two (distinct) key pairs, which are used in an ad-hoc network and in the infra network separately. For a particular signer, pick $x_2 \in_R \mathbb{Z}_q$ at random, and compute $y_2 \leftarrow g_1^{x_2}$. The infra secret key is $x_2 \in \mathbb{Z}_q$ and the corresponding public key is $y_2 \in \mathcal{G}_1$. Then, the ad-hoc key and trusted notary's key pairs, (x_1, y_1) and (x_N, y_N), are generated, respectively, in the same manner as the infra key generation. Note that an ad-hoc key might be linearly independent of the infra secret.

Signature generation. Given the key pairs (x_1, y_1) and (x_2, y_2), and the trusted notary's public key y_N, the algorithm to sign a message $M \in \{0,1\}^*$ runs as follows. The signer chooses $r_1, r_2, k_1, k_2 \in_R \mathbb{Z}_q$ at random, and computes

$$\begin{gathered} h \leftarrow H_1(M, r_1),\ \sigma_1 \leftarrow h^{x_1},\ \sigma_2 \leftarrow h^{x_2},\ u_1 \leftarrow g_2^{k_1},\ v_1 \leftarrow h^{k_1},\ u_2 \leftarrow g_2^{k_2},\ v_2 \leftarrow h^{k_2}, \\ c \leftarrow H_2(h, \psi(y_1), \psi(y_2), \sigma_1, \sigma_2, u_1, v_1, u_2, v_2),\ s_1 \leftarrow k_1 + cx_1,\ s_2 \leftarrow k_2 + cx_2, \\ w = \langle\ g_2^{r_2}, \sigma_2 \cdot \psi(y_N)^{r_2}\ \rangle\,. \end{gathered}$$

The P2DL signature is $\Sigma = (r_1, \sigma_1, w, s_1, s_2, c)$. This will be sent to the verifier along with the message M and *implicitly* public keys y_1, y_2, and y_N.

At first glance, we can see two individual co-GDH signatures σ_1, σ_2 generated under the ad-hoc and infra secrets, respectively. However, the signer does not want any recipient to "instantly" possess her infra signature σ_2 until the trusted notary permits publication of σ_2. Thus, σ_2 is encrypted into w with the notary's public key, while σ_1 is not. The ciphertext w is exactly in the form of an ElGamal type ciphertext [ElG85]. In addition, a non-interactive zero-knowledge argument proving that $(y_1, \sigma_1) \in \mathsf{EDL}(g_1, h)$ and, simultaneously, $(y_2, \sigma_2) \in \mathsf{EDL}(g_1, h)$ is presented to demonstrate the signer's possession of two discrete logarithms x_1, x_2. Indeed, this ZK argument works as a nexus or correlation between σ_1 and σ_2.

Verification. Upon receipt of (M, Σ), any user can validate σ_1 and check if the ad-hoc public key of the signer is revoked. First, a verifier parses $\Sigma \overset{\mathrm{P}}{\to} r_1 \| \sigma_1 \| w \| s_1 \| s_2 \| c$. The verifier then computes $h \leftarrow H_1(M, r_1)$ and checks if

$$\hat{e}(g_1, \sigma_1) \stackrel{?}{=} \hat{e}(y_1, h). \tag{1}$$

Consistency is easily proved because $\hat{e}(g_1, \sigma_1) = \hat{e}(g_1, h)^{x_1} = \hat{e}(g_1^{x_1}, h) = \hat{e}(y_1, h)$. However, the infra signature σ_2 cannot be verified instantly because it is encrypted. One can verify the *encrypted* signature instead since the encrypted infra signature in our scheme is analogous to the bilinear verifiably-encrypted signature.[2] Perform the following: parse $w \to U \| V$; check if the equation below holds.

$$\hat{e}(g_1, V) \stackrel{?}{=} \hat{e}(y_2, h) \cdot \hat{e}(y_N, U) \tag{2}$$

[2] The bilinear verifiably-encrypted signature [BGLS03] is a simple combination of the co-GDH signature and ElGamal cryptosystem. Our design goal is to provide the universal verifiability of digital signatures across two domains, while the verifiably-encrypted signature is applicable to the contract signing protocol with a single domain setting. Moreover, the resulting signature scheme features a *tight* security reduction.

This also ensures consistency of verification for any valid encrypted signature; $\hat{e}(g_1, V) = \hat{e}(g_1, \sigma_2 \cdot \psi(y_N)^{r_2}) = \hat{e}(g_1, h^{x_2}) \cdot \hat{e}(g_1, \psi(y_N)^{r_2}) = \hat{e}(y_2, h) \cdot \hat{e}(g_1, U^{x_N}) = \hat{e}(y_2, h) \cdot \hat{e}(y_N, U)$.

The verifier, voluntarily, forwards (M, Σ) to the trusted notary when he or she wishes to confirm the authorship of the ad-hoc signature σ_1 and to obtain a "translation" of σ_1 available in the infra network. The notary in turn responds with the validity of the signature and the infra version of the given signature i.e., σ_2 as required. We name this algorithm *Notarization*.

Notarization. The notarization algorithm takes as input the notary's decryption key x_N, the signer's ad-hoc and infra public keys y_1, y_2, and (M, Σ). After parsing the $\mathsf{P2DL}$ signature, the notary performs the following:

- Check whether the public key y_2 is revoked (e.g, OCSP [MAM+99] and CRL profile [HFPS99]); If so, output $\perp$ and terminate;
- Otherwise, parse the ciphertext into two components such that $w \stackrel{\mathrm{P}}{\rightarrow} U \| V$;
- Decrypt the ciphertext: $\sigma_2 \leftarrow V / U^{x_N}$;
- Compute $h \leftarrow H_1(M, r_1)$;
- Compute $u_1 \leftarrow g_2^{s_1} \psi(y_1)^{-c}$, $\;v_1 \leftarrow h^{s_1} \sigma_1^{-c}$, $\;u_2 \leftarrow g_2^{s_2} \psi(y_2)^{-c}$, $\;v_2 \leftarrow h^{s_2} \sigma_2^{-c}$;
- Compute $c' \leftarrow H_2(h, \psi(y_1), \psi(y_2), \sigma_1, \sigma_2, u_1, v_1, u_2, v_2)$;
- If $c' = c$, publish σ_2 as the infra signature; Otherwise, output $\perp$.

This algorithm runs when the verifier has made a request to the trusted notary for the aforementioned reasons. The notarization algorithm consists of two types of checking procedures. First, it checks whether the signer's infra public key is revoked. Second, the notary checks if the proof system assures that $(y_1, \sigma_1) \in \mathsf{EDL}(g_1, h) \wedge (y_2, \sigma_2) \in \mathsf{EDL}(g_1, h)$. Equivalently, if the zero-knowledge is proved, it convinces the notary that the signer owns both $\mathtt{dlog}_{g_1} y_1$ and $\mathtt{dlog}_{g_1} y_2$. If all the checking procedures output "YES," the notary determines to publish σ_2 as the rendering of σ_1 on behalf of the signer. Alternatively, the trusted notary appends its signature on this infra signature σ_2 to enable every user to distinguish the notary's publications from ordinary (infra) signatures.

Confirmation. Upon satisfaction of notarization, the verifier receives an infra signature σ_2 from the notary. The verifier checks if the following equation holds true

$$\hat{e}(g_1, \sigma_2) \stackrel{?}{=} \hat{e}(y_2, h). \tag{3}$$

The signature is verified under the signer's public key, which has already been certified. The consistency in verifying σ_2 is easily provable as we have likewise done for (1).

4 Security Analysis

4.1 Proving Possession of Two Discrete Logarithms

The zero-knowledge proof of the equality of two discrete logarithms introduced in Sect 2.4 helps a prover to convince a verifier of his possession of a *single* secret exponent $x \in \mathbb{Z}_q$

that is the secret key of the corresponding public key y and the discrete logarithm of another element σ with respect to the base h as well: $x = \mathtt{dlog}_g y = \mathtt{dlog}_h \sigma$. The $\mathsf{P2DL}$ signature scheme, however, requires a non-interactive zero-knowledge proof of the possession of two (not single but distinct) discrete logarithms x_1 and x_2, which is an extension of the zero-knowledge proof of the equality of two discrete logarithms. We refer to this special honest-verifier proof system as $\mathsf{P2DL}$.

The $\mathsf{P2DL}$ proof system takes as input the prover's secret exponents x_1 and x_2 as elements of a certain group $\mathcal{G}$, two generators g and h of $\mathcal{G}$, a hash function H_2 that maps from $(\mathcal{G})^9$ to $\mathbb{Z}_q$. First, the prover picks $k_1, k_2 \in_R \mathbb{Z}_q$ at random. Next, proceed the following in order to prove that $(y_1, \sigma_1) \in \mathsf{EDL}(g, h) \wedge (y_2, \sigma_2) \in \mathsf{EDL}(g, h)$:

$$\begin{gathered} u_1 \leftarrow g^{k_1},\ v_1 \leftarrow h^{k_1},\ u_2 \leftarrow g^{k_2},\ v_2 \leftarrow h^{k_2}, \\ c \leftarrow H_2(h, y_1, y_2, \sigma_1, \sigma_2, u_1, v_1, u_2, v_2), \\ s_1 \leftarrow k_1 + cx_1,\ s_2 \leftarrow k_2 + cx_2. \end{gathered}$$

The binary operator "$\wedge$" implies the logical operator "AND." The resulting proof is (c, s_1, s_2) and can be verified by first reconstructing the commitments

$$u_1' \leftarrow g^{s_1} y_1^{-c},\ v_1' \leftarrow h^{s_1} \sigma_1^{-c},\ u_2' \leftarrow g^{s_2} y_2^{-c},\ v_2' \leftarrow h^{s_2} \sigma_2^{-c},$$

where $y_1 = g^{x_1}$ and $y_2 = g^{x_2}$, and then checking the equations below

$$c' \leftarrow H_2(h, y_1, y_2, \sigma_1, \sigma_2, u_1', v_1', u_2', v_2') \quad \text{and} \quad c' \stackrel{?}{=} c.$$

The verifier accepts if $c' = c$; otherwise, the verifier rejects.

Theorem 1. *The P2DL ZK protocol mentioned above is a special honest-verifier proof system for proving that $(y_1, \sigma_1) \in \mathsf{EDL}(g, h) \wedge (y_2, \sigma_2) \in \mathsf{EDL}(g, h)$, assuming the couple of keys belongs to a user.*

Proof. Completeness of the protocol can be easily seen so that an honest prover will always succeed in constructing a valid proof since

$$\begin{gathered} u_1' = g^{s_1} y_1^{-c} = g^{k_1 + cx_1} g^{-cx_1} = g^{k_1} = u_1, \\ v_1' = h^{s_1} \sigma_1^{-c} = h^{k_1 + cx_1} h^{-cx_1} = h^{k_1} = v_1. \end{gathered}$$

Likewise, we can see the correctness for u_2' and v_2'. Therefore, for an honest prover, $c' = c$. Let us now consider the soundness of $\mathsf{P2DL}$ ZK protocol. Suppose that a cheating prover who knows neither x_1 nor x_2 can generate another correct response and challenge pair $(\hat{c}, s_1', s_2')$ that differs from the given (c, s_1, s_2), then

$$\begin{cases} g^{s_1 - s_1'} = y_1^{\hat{c} - c}, & g^{s_2 - s_2'} = y_2^{\hat{c} - c}, \\ h^{s_1 - s_1'} = \sigma_1^{\hat{c} - c}, & h^{s_2 - s_2'} = \sigma_2^{\hat{c} - c}, \end{cases}$$

and hence

$$\begin{cases} \mathtt{dlog}_g y_1 = \mathtt{dlog}_h \sigma_1 = \frac{s_1 - s_1'}{\hat{c} - c}, \\ \mathtt{dlog}_g y_2 = \mathtt{dlog}_h \sigma_2 = \frac{s_2 - s_2'}{\hat{c} - c}. \end{cases}$$

This contradicts the assumption that the cheating prover knows neither x_1 nor x_2. Thus, the soundness probability is at most $1/q^2$. Even when a prover knows one of the discrete logarithms, the prover's probability of successfully cheating is at most approximately $1/q$. $\square$

In the actual run of P2DL signature scheme, a slightly modified version of this proof system is used, where an isomorphism ψ and two distinct groups $\mathcal{G}_1 = \langle g_1 \rangle$ and $\mathcal{G}_2 = \langle g_2 \rangle$ with a prime order q might be used.

4.2 Security against Existential Forgeries

Boneh *et al.* have shown that the co-GDH signature scheme is secure, namely existentially unforgeable, under a chosen message attack as long as the co-GDH assumption holds. The P2DL signature introduced here uses the co-GDH signature as the underlying signature scheme. However, it still remains unproven that an attacker who has at most one of the secret keys x_1 and x_2 cannot generate any valid P2DL signature tuple $\Sigma = (r_1, \sigma_1, w, s_1, s_2, c)$ with a non-negligible probability.

Theorem 2. *The P2DL signature scheme is provably secure against existential forgery under adaptive chosen message attacks assuming that (a) all the hash functions are chosen from the random oracles, and (b) the co-GDH assumption holds.*

The proof of this theorem is described in the appendix A.

5 Concluding Remarks

Coalition-Resistance. Observe that a colluding couple of ad-hoc and infra users can generate a valid (but forged) P2DL signature since the couple of used key pairs for ad-hoc and infra users is completely *loosely-coupled*. Therefore, we should make any relationship between the signing keys. A simple countermeasure we have in mind is to provide an additional administration policy on issuing ad-hoc certificates. The signer A has the following formatted certificate, which is denoted by

$$\textsc{Cert}_A(\text{ad-hoc}) = \langle ID_1, PK_1, \textsc{Cert}_A(\text{infra}), \mathsf{params}, Sig_{\mathrm{CA}_1}(\text{all parts thereof})\rangle$$

where $ID_1 = ID_A(\text{ad-hoc})$ and $PK_1 = PK_A(\text{ad-hoc})$. In addition, params represents the domain parameters $(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T, g_1, g_2, \psi, \hat{e}, q)$ and $Sig_{\mathrm{CA}_1}(\cdot)$ denotes an (ordinary) signature of the ad-hoc CA. Whenever issuing an ad-hoc certificate, it is mandatory for the ad-hoc CA to authenticate whether the user is the owner of the predetermined infra certificate

$$\textsc{Cert}_A(\text{infra}) = \langle ID_2, PK_2, \mathsf{params}, Sig_{\mathrm{CA}_2}(\text{all parts thereof})\rangle.$$

Each of ID_2, PK_2, and $Sig_{\mathrm{CA}_2}(\cdot)$ has the equivalent meaning to ID_1, PK_1, and $Sig_{\mathrm{CA}_1}(\cdot)$ in the infra network. Note that the same domain parameters params are used even in the infra network. Achieving the coalition-resistance via more sophisticated constructions such as group signatures [ACJT00, BBS04] can be a compelling approach, but it is not of our interest.

Towards Heterogeneous Settings. Our protocol is constructed only within a "homogeneous" mathematical setting. In other words, both in ad-hoc network and in infra network, the same domain parameters and mathematical functions are appropriately assumed. Nonetheless, it would be more preferable to consider the universal verifiability among "heterogeneous" domains with distinct domain parameters to achieve the so-called *ubiquitous* security. As far as we know, this generalized construction still remains unanswered.

References

[ACJT00] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology – CRYPTO '00*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer-Verlag, August 2000. 12

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology – CRYPTO '04*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer-Verlag, August 2004. 12

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology – EUROCRYPT '03*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer-Verlag, May 2003. 3, 7, 9

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 17(4):297–319, September 2004. 5, 7

[Bon98] Dan Boneh. The decision Diffie-Hellman problem. In *Proceedings of the Third Algorithmic Number Theory Symposium*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer-Verlag, 1998. 6

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of ACM Conference on Computer and Communications Security '93*, pages 62–73. ACM Press, 1993. 3, 8

[CM05] Benoît Chevallier-Mames. An efficient CDH-based signature scheme with a tight security reduction. In *Advances in Cryptology – CRYPTO '05*, volume 3621 of *Lecture Notes in Computer Science*, pages 511–526. Springer-Verlag, August 2005. 1, 8

[CP93] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In *Advances in Cryptology – CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer-Verlag, August 1993. 3, 7, 8

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976. 6

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, July 1985. 9

[GJ03] Eu-Jin Goh and Stanisław Jareki. A signature scheme as secure as the Diffie-Hellman problem. In *Advances in Cryptology – EUROCRYPT '03*, volume 2656 of *Lecture Notes in Computer Science*, pages 401–415. Springer-Verlag, May 2003. 1, 8, 19

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988. 7

[HFPS99] R. Housley, W. Ford, W. Polk, and D. Solo. Certificate and CRL profile. RFC 2459, January 1999. `http://www.ietf.org/rfc/rfc2459.txt`. 10

[ITU97] X.509 (1997 e): Information Technology – Open Systems Interconnection – The directory: Authentication framework, 1997. 3, 4

[JN01] Antoine Joux and Kim Nguyen. Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Cryptology ePrint Archive, Report 2001/003, 2001. `http://eprint.iacr.org/`. 7

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In *Proceedings of ACM Conference on Computer and Communications Security '03*, pages 155–164. ACM Press, 2003. 8

[LOM05] KyungKeun Lee, JoongHyo Oh, and SangJae Moon. How to generate universally verifiable signatures in ad-hoc networks. In *Secure Mobile Ad-hoc Networks and Sensors Workshop – MADNES '05*. will be published in LNCS, Springer-Verlag, September 2005. 1

[MAM+99] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. Online certificate status protocol - OCSP. RFC 2560, 1999. `http://www.ietf.org/rfc/rfc2560.txt`. 10

[MW99] 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. 6

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. In *International Workshop on Practice and Theory in Public Key Cryptography – PKC'01*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118. Springer-Verlag, February 2001. 3, 7

[SG98] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *Advances in Cryptology – EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, May 1998. 8

[ZH99] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, November-December 1999. 3

A Proof of Theorem 2

Suppose that a forger algorithm $\mathcal{F}$ $(t, q_H, q_S, q_N, \epsilon)$-breaks the P2DL signature scheme. We construct a simulator algorithm $\mathcal{S}$, which takes at least one of an ad-hoc key y_1 and infra key y_2 as input. The simulator $\mathcal{S}$ works in order to forge a co-GDH signature or to directly solve the CDH problem in group $\mathcal{G}_1$. Thus, we subdivide the simulator $\mathcal{S}$ into $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ as follows.

- $\mathcal{S}_1$ attempts to forge a co-GDH signature σ_1 that is the resulting ad-hoc signature from the forger $\mathcal{F}$ algorithm against the P2DL scheme. The input to the simulator is y_1.
- $\mathcal{S}_2$ attempts to forge a co-GDH signature σ_2 that is the resulting infra signature from the forger algorithm $\mathcal{F}$ against the P2DL scheme. $\mathcal{S}_2$ takes y_2 as input.
- $\mathcal{S}_3$ attempts to solve a CDH problem given g_1, g_1^a, and g_1^b as input, without knowing any secret exponent $a \in \mathbb{Z}_q$ or $b \in \mathbb{Z}_q$. The input to $\mathcal{S}_3$ is a tuple (g_1, g_1^a, g_1^b).

The theorem now follows immediately from the following three lemmas.

Lemma 3 (Ad-Hoc Signature Unforgeability)**.** *If the co-GDH signature scheme is $(t', q'_H, q'_S, \epsilon')$-secure against existential forgery on the groups $\mathcal{G}_1$ and $\mathcal{G}_2$ where these groups are co-GDH groups of prime order q with an isomorphism $\psi : \mathcal{G}_1 \to \mathcal{G}_2$ such that $\psi(g_1) = g_2$ and a bilinear map $\hat{e} : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$, then the P2DL signature scheme is $(t, q_H, q_S, q_N, \epsilon)$-secure against existential (ad-hoc signature) forgery under adaptive chosen message attacks in the random oracle model, where*

$$\epsilon \le \epsilon' + q_S \cdot q_H \cdot q^{-3} \tag{4}$$

$$t \approx t' - 15 \cdot q_S \cdot t_{\exp} \tag{5}$$

Proof. The cost of a multi-base exponentiation is estimated to be about 20% more than the cost of a single exponentiation. We denotes $t_{\exp}$ the running time of a single exponentiation in each of defined groups $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_T$.

Given an ad-hoc public key y_1, algorithm $\mathcal{S}_1$ simulates a run of the P2DL signature scheme to the forger $\mathcal{F}$. As a simulator, $\mathcal{S}_1$ answers $\mathcal{F}$'s hash function queries, signature oracle queries, notarization oracle queries, and it tries to translate $\mathcal{F}$'s possible forgery Σ into a co-GDH signature forgery σ_1 of y_1. Two types of signature oracles are presented; one belongs to the co-GDH challenger and the other belongs to $\mathcal{S}_1$ under the simulator's control. Algorithm $\mathcal{S}_1$ simulates the challenger and interacts with $\mathcal{F}$ as follows.

Setup. The simulator $\mathcal{S}_1$ generates two key pairs (x_2, y_2) and (x_N, y_N) running the Key Generation algorithm described in Sect 3. The key pair (x_2, y_2) serves as an infra key pair, and meanwhile (x_N, y_N) works as the trusted notary's key pair. Now, $\mathcal{S}_1$ runs $\mathcal{F}$ providing the public keys y_1, y_2, and y_N, and public parameters $(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T, g_1, g_2, \psi, \hat{e}, q)$ as input.

$\mathbf{H_1}$-oracle Queries. If the forger $\mathcal{F}$ requests a hash query on a string (M, r_1), algorithm $\mathcal{S}_1$ makes a query on (M, r_1) to the challenger's hash oracle, receiving some value $h \leftarrow H_1(M, r_1)$. Then $\mathcal{S}_1$ simply forwards h to $\mathcal{F}$.

$\mathbf{H_2}$-oracle Queries. $\mathcal{S}_3$ answers with a random integer in $\mathbb{Z}_q$.

Signature Queries. The forger $\mathcal{F}$ asks for a P2DL signature on message M. Algorithm $\mathcal{S}_1$ has to create a valid signature tuple without knowing the ad-hoc secret key x_1 even though $\mathcal{S}_1$ has access to a signing oracle for y_1 and a H_1 hash oracle belong to the challenger. $\mathcal{S}_1$ proceeds as follows:

1. $\mathcal{S}_1$ randomly chooses $r_1 \in_R \mathbb{Z}_q$.
2. $\mathcal{S}_1$ asks a H_1 query on (M, r_1), receiving some value $h \leftarrow H_1(M, r_1)$ and creates $\sigma_2 \leftarrow h^{x_2}$ as an infra signature.
3. $\mathcal{S}_1$ queries the challenger's signing oracle for y_1 on $M \| r_1$. The signing oracle will answer $\sigma_1 \leftarrow h^{x_1}$ to $\mathcal{S}_1$.
4. Thereafter, $\mathcal{S}_1$ simulates the P2DL ZK protocol as such: picks at random $\kappa, c, s_1 \in_R \mathbb{Z}_q$, computes $u_2 \leftarrow g_2^{\kappa}$ and $v_2 \leftarrow h^{\kappa}$, computes $s_2 \leftarrow \kappa + cx_2$, and assigns $u_1 \leftarrow g_2^{s_1}\psi(y_1)^{-c}$ and $v_1 \leftarrow h^{s_1}\sigma_1^{-c}$.
5. If H_2 has ever been queried on inputs $(h, \psi(y_1), \psi(y_2), \sigma_1, \sigma_2, u_1, v_1, u_2, v_2)$ before, $\mathcal{S}_1$ aborts. Otherwise, $H_2(h, \psi(y_1), \psi(y_2), \sigma_1, \sigma_2, u_1, v_1, u_2, v_2) \triangleq c$.
6. $\mathcal{S}_1$ selects $r_2 \in_R \mathbb{Z}_q$ at random, and encrypts σ_2 into $\langle g_2^{r_2}, \sigma_2 \cdot \psi(y_N)^{r_2} \rangle$ under the notary's key y_N. Finally, $\mathcal{S}_1$ returns $\Sigma = (r_1, \sigma_1, w, s_1, s_2, c)$ as the P2DL signature of M.

Notarization Queries. When the forger $\mathcal{F}$ requests notarization for (M, Σ), $\mathcal{S}_1$ checks whether the given signature $\Sigma = (r_1, \sigma_1, w, s_1, s_2, c)$ is valid and, if so, returns $\sigma_2 \leftarrow V/U^{x_N}$, parsing the ciphertext $w \xrightarrow{\mathrm{P}} U \| V$.

Guess. If $\mathcal{F}$ provides a valid P2DL signature pair (M^*, Σ^*) for some previously unsigned non-trivial message M^*, then algorithm $\mathcal{S}_1$ extracts σ_1^* from $\Sigma^* = (r_1^*, \sigma_1^*, w^*, s_1^*, s_2^*, c^*)$. Finally, $\mathcal{S}_1$ outputs σ_1^* as a co-GDH signature forgery on a message $M^* \| r_1^*$.

□

Claim 1. *If algorithm $\mathcal{S}_1$ does not abort during the simulation, the forger algorithm $\mathcal{F}$'s view is identical to its view in the real attack.*

Proof of claim. The responses to H_1 and H_2 queries are as in the actual attack since each response is uniformly and independently distributed in the probability spaces $\mathcal{G}_2$ and $\mathbb{Z}_q$, respectively. As a consequence, the success probability to forge a co-GDH signature is equal to the probability of forging a P2DL signature.

□

Remark 1. The running time of algorithm $\mathcal{S}_1$ can be calculated by inspection: In Setup phase, the simulator computes two single exponentiations. Each query to the signing oracle requires six single exponentiations together with two multi-base exponentiations. Each query to the notarization oracle requires two single exponentiations and four multi-base exponentiations. Assuming that $q_S \approx q_N$, we get the approximation of the running time in (5).

It remains to bound the probability that the simulator $\mathcal{S}_1$ aborts during simulation. Such an event is defined as below:

$\mathcal{E}$ is the event that the same input to the H_2-oracle is queried one more time at the fifth step of the signature oracle simulation.

Claim 2. $\mathsf{Pr}[\mathcal{E}] \leq q_S \cdot (q_H + q_S) \cdot q^{-3}$

Proof of claim. To estimate the probability of event $\mathcal{E}$, let us take the input to the H_2-oracle into account. The input tuple is represented as a sequence of several strings: $(h, \psi(y_1), \psi(y_2), \sigma_1, \sigma_2, u_1, v_1, u_2, v_2)$. For simplicity, we ignore the isomorphism ψ but it makes no serious change in security. Hence we have $(h, y_1, y_2, \sigma_1, \sigma_2, u_1, v_1, u_2, v_2)$, which can be represented as $(g_2^\alpha, y_1, y_2, y_1^\alpha, y_2^\alpha, g_2^\beta, g_2^{\alpha\beta}, g_2^\gamma, g_2^{\alpha\gamma})$, where all the exponents are considered as random coins as such $(\alpha, \beta, \gamma) \in_R \mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q$. Therefore, we obtain the overall probability of collision is less than $q_S \cdot (q_H + q_S) \cdot q^{-3}$ for the simulation of the signature oracle.

□

Remark 2. Reasonably assuming $q_H \gg q_S$, we see that $\mathcal{S}_1$ successfully forges a co-GDH signature with an advantage of (4). Since the amount $q_S \cdot q_H \cdot q^{-3}$ is enough small to be negligible, forging a P2DL signature with a single secret key is identical to forging a co-GDH signature as required.

Lemma 4 (Infra Signature Unforgeability)**.** *If the co-GDH signature scheme is $(t', q'_H, q'_S, \epsilon')$-secure against existential forgery on the same groups $\mathcal{G}_1$ and $\mathcal{G}_2$ as defined in Lemma 3, then the P2DL signature scheme is $(t, q_H, q_S, q_N, \epsilon)$-secure against existential (infra signature) forgery under adaptive chosen message attacks in the random oracle model.*

Proof. Simply substituting the input to $\mathcal{S}_1$ for y_2, $\mathcal{S}_2$ obtains exactly the same result as Lemma 3. □

In the following lemma, we reduce the successful forgery against the P2DL scheme to the CDH problem solver in $\mathcal{G}_1$.

Lemma 5 (Unforgeability against Attackers with No Secret)**.** *Suppose that $\mathcal{G}_1$ is a (t', ϵ')-CDH group that forms a co-GDH group pair with $\mathcal{G}_2$ as defined above. Then the P2DL signature scheme is $(t, q_H, q_S, q_N, \epsilon)$-secure against existential forgery on adaptive chosen message attacks in the random oracle model, where*

$$\epsilon \leq \epsilon' + q_S \cdot q_H \cdot q^{-1} \tag{6}$$

$$t \approx t' - (q_H + 17 \cdot q_S) \cdot t_{\exp} \tag{7}$$

Proof. Given three elements g_1, g_1^a, and g_1^b in group $\mathcal{G}_1$, algorithm $\mathcal{S}_3$ simulates a run of the P2DL signature scheme to the forger $\mathcal{F}$. The simulator $\mathcal{S}_3$ answers $\mathcal{F}$'s H_1 and H_2 hash queries, the P2DL signature oracle queries, notarization oracle queries, and it tries to translate $\mathcal{F}$'s possible forgery Σ into an answer to the CDH problem g_1^{ab}. Algorithm $\mathcal{S}_3$ simulates the challenger and interacts with $\mathcal{F}$ as follows.

Setup. The simulator $\mathcal{S}_3$ assigns $y_1 \triangleq g_1^a$ and $y_2 \triangleq g_1^b$ and generates a notary's key pair (x_N, y_N). Now, $\mathcal{S}_3$ runs $\mathcal{F}$ providing the public keys y_1, y_2, and y_N, and public parameters $(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T, g_1, g_2, \psi, \hat{e}, q)$ as input.

$\mathbf{H_1}$-oracle Queries. When the forger $\mathcal{F}$ asks a hash query on a string (M, r_1), $\mathcal{S}_3$ replies with $H_1(M, r_1) \leftarrow (\psi(y_2))^d$, selecting at random $d \in_R \mathbb{Z}_q$.

$\mathbf{H_2}$-oracle Queries. Handled as in the Lemma 3.

Signature Queries. The simulator $\mathcal{S}_3$ performs as follows when $\mathcal{F}$ requests a $\mathsf{P2DL}$ signature query on message M.

1. $\mathcal{S}_3$ randomly chooses $r_1 \in_R \mathbb{Z}_q$. If the same H_1-oracle query on (M, r_1) has been granted, the simulator aborts.
2. Otherwise, $\mathcal{S}_3$ assigns $h \leftarrow g_2^\kappa$, $\sigma_1 \leftarrow \psi(y_1)^\kappa$, and $\sigma_2 \leftarrow \psi(y_2)^\kappa$ by picking $\kappa \in_R \mathbb{Z}_q$ at random.
3. Next $\mathcal{S}_3$ simulates the $\mathsf{P2DL}$ ZK protocol as such: picks $c, s_1, s_2 \in_R \mathbb{Z}_q$, and assigns $u_1 \leftarrow g_2^{s_1}\psi(y_1)^{-c}$, $v_1 \leftarrow h^{s_1}\sigma_1^{-c}$, $u_2 \leftarrow g_2^{s_2}\psi(y_2)^{-c}$, and $v_2 \leftarrow h^{s_2}\sigma_2^{-c}$.
4. If H_2 has ever been queried on inputs $(h, \psi(y_1), \psi(y_2), \sigma_1, \sigma_2, u_1, v_1, u_2, v_2)$ before, $\mathcal{S}_3$ aborts. Otherwise, $H_2(h, \psi(y_1), \psi(y_2), \sigma_1, \sigma_2, u_1, v_1, u_2, v_2) \triangleq c$.
5. $\mathcal{S}_3$ selects $r_2 \in_R \mathbb{Z}_q$ at random, and encrypts σ_2 into $\langle g_2^{r_2}, \sigma_2 \cdot \psi(y_N)^{r_2} \rangle$ under the notary's key y_N. Finally, $\mathcal{S}_3$ returns $\Sigma = (r_1, \sigma_1, w, s_1, s_2, c)$ as the $\mathsf{P2DL}$ signature of M.

Notarization Queries. Handled as in the Lemma 3.

Guess. If $\mathcal{F}$ provides a valid $\mathsf{P2DL}$ signature pair (M^*, Σ^*) for some previously unsigned non-trivial message M^*, and then algorithm $\mathcal{S}_3$ extracts σ_1^* from $\Sigma^* = (r_1^*, \sigma_1^*, w^*, s_1^*, s_2^*, c^*)$. If (M^*, r_1^*) has not been queried to the H_1-oracle before, the simulator $\mathcal{S}_3$ aborts. Otherwise, $\mathcal{S}_3$ computes $(\sigma_1)^{1/d}$. With the inverse of a given isomorphism, $\psi^{-1}\left((\sigma_1)^{1/d}\right)$ is the answer to the CDH question. Finally, $\mathcal{S}_3$ succeeds in computing $(g_1)^{ab}$.

□

Claim 3. *If algorithm $\mathcal{S}_3$ does not abort during the simulation, the forger algorithm $\mathcal{F}$'s view is identical to its view in the real attack.*

Proof of claim. The responses to H_1 and H_2-queries are identical to those in the actual attack since each response is uniformly and independently distributed in the probability spaces $\mathcal{G}_2$ and $\mathbb{Z}_q$, respectively. In particular, the signing oracle creates a valid $\mathsf{P2DL}$ signature for any message. Indeed, any output of $\mathcal{S}_3$'s signing oracle is always valid. Hence, the success probability in solving a CDH challenge is equal to the probability of forging a $\mathsf{P2DL}$ signature. □

Remark 3. The running time of algorithm $\mathcal{S}_3$ can be estimated by summing up the following: In Setup phase and H_1-oracle simulation, the simulator computes at most $q_H + 1$ exponentiations. Each query to the signing oracle requires five single exponentiations together with four multi-base exponentiations. Each query to the notarization oracle requires two single exponentiations and four multi-base exponentiations.

$$(q_H + 9.8 \cdot q_S + 6.8 \cdot q_N + 1) \cdot t_{\exp}$$

Assuming that $q_S \approx q_N \gg 1$, we obtain approximately (7).

To complete the proof of Lemma 5, it remains to upper bound the probability that the simulator $\mathcal{S}_3$ aborts during simulation. We define several relevant events:

$\mathcal{E}_1$ is the event that the signing oracle of $\mathcal{S}_3$ aborts at the first step answering signature queries.

$\mathcal{E}_2$ is the event that the same input to the H_2-oracle is queried one more time at the fourth step of the signature oracle simulation.

$\mathcal{E}_3$ is the event that the simulator $\mathcal{S}_3$ aborts at the Guess stage when (M^*, r_1^*) of the resulting $\mathcal{F}$'s output is "not" found in the history of H_1-oracle queries.

Claim 4. $\mathsf{Pr}[\mathcal{E}_1 \vee \mathcal{E}_2 \vee \mathcal{E}_3] \;\leq\; q_S \cdot (q_H + q_S) \cdot q^{-1} + q_S \cdot (q_H + q_S) \cdot q^{-3} + q^{-1}$

Proof of claim. Observe that $\mathsf{Pr}[\mathcal{E}_1 \vee \mathcal{E}_2 \vee \mathcal{E}_3] \leq \mathsf{Pr}[\mathcal{E}_1] + \mathsf{Pr}[\mathcal{E}_2] + \mathsf{Pr}[\mathcal{E}_3]$. First, the event $\mathcal{E}_1$ occurs when simulator $\mathcal{S}_1$ chooses an r_1 that was a previously given input to the H_1-oracle. For q_S signature queries, the termination probability with failure is then at most $q_S \cdot (q_H + q_S) \cdot q^{-1}$. Second, the event $\mathcal{E}_2$ is regarded as the same event we considered in Claim 2. Lastly, the event $\mathcal{E}_3$ occurs when the forger $\mathcal{F}$ provides a signature satisfying $\sigma_1^{1/x_1} = h$ on its successful forgery. This probability becomes at most q^{-1}.

Unlike in [GJ03], we do not have to deliberate an extraordinary event of which $\mathcal{F}$ outputs a valid forgery even when $(y_1, \sigma_1) \notin \mathsf{EDL}(g_1, h)$. In our attack game, such an event does not occur because a valid output of the forger $\mathcal{F}$ always satisfies that $\hat{e}(g_1, \sigma_1) = \hat{e}(y_1, h)$.

□

Remark 4. Reasonably assuming $q_H \gg q_S \gg 1$, we see that $\mathcal{S}_3$ successfully solves a CDH problem in $\mathcal{G}_1$ with an advantage of (6).