The Security of the FDH Variant of Chaum's Undeniable Signature Scheme ⋆

Wakaha Ogata[1] and Kaoru Kurosawa[2] and Swee-Huay Heng[3]

[1] Tokyo Institute of Technology,
2-12-1 O-okayama, Meguro-ku,Tokyo, 152-8552 Japan
wakaha@craft.titech.ac.jp

[2] Department of Computer and Information Sciences,
Ibaraki University,
4-12-1 Nakanarusawa, Hitachi, Ibaraki 316-8511, Japan
kurosawa@cis.ibaraki.ac.jp

[3] Multimedia University,
Jalan Ayer Keroh Lama, 75450 Melaka, Malaysia
shheng@mmu.edu.my

Abstract. In this paper, we first introduce a new kind of adversarial goal called *forge-and-impersonate* in undeniable signature schemes. Note that forgeability does not necessarily imply impersonation ability. We then classify the security of the FDH variant of Chaum's undeniable signature scheme according to three dimensions, the goal of adversaries, the attacks and the ZK level of confirmation and disavowal protocols. We finally relate each security to some well-known computational problem. In particular, we prove that the security of the FDH variant of Chaum's scheme with NIZK confirmation and disavowal protocols is equivalent to the CDH problem, as opposed to the GDH problem as claimed by Okamoto and Pointcheval.

Keywords: Undeniable signature, security analysis

1 Introduction

1.1 Background

The notion of undeniable signature schemes was introduced by Chaum and van Antwerpen in 1989 [11]. Since then, there have been a wide range of research covering a variety of different schemes for undeniable signatures. The validity or invalidity of an undeniable signature can only be verified with the signer's consent by engaging interactively or non-interactively in a confirmation or disavowal protocol respectively, as opposed to a digital signature in which its validity is universally verifiable. Extended schemes possess variable degrees of security and additional features such as convertibility [6, 15, 23], designated-verifier technique

⋆ The proceedings version of this paper was presented at PKC 2005. This is an extended version.

[21], designated-confirmer technique [9], and so on. Among others, we also include [8, 12, 19, 18, 17].

Undeniable signatures have various applications in cryptography such as in licensing softwares, electronic voting and auctions. The most popular application is in licensing softwares. For example, software vendors might want to sign on their products to provide authenticity to their paying customers. Nevertheless, they strictly disallow dishonest users who have illegally duplicated their softwares to verify the validity of these signatures. Undeniable signature scheme plays an important role here as it allows only legitimate users to verify the validity of the signatures on the softwares.

The first proposal of undeniable signature which is based on the intractability of the computational Diffie-Hellman (CDH) problem was due to Chaum and van Antwerpen [11] and it was further improved by Chaum [8]. It is a simple and nice scheme.

On the other hand, in general, each undeniable signature scheme may have three variants of confirmation and disavowal protocols, namely, the perfect zero-knowledge protocol (ZKIP), the 3-move honest-verifier zero-knowledge protocol (HVZK) and the non-interactive zero-knowledge protocol (NIZK) with designated-verifier technique.

However, the unforgeability of Chaum's undeniable signature scheme (under any types of confirmation and disavowal protocols) has been an open problem for a long time. Recently, Okamoto and Pointcheval [24] considered the security of the full-domain hash (FDH) [5, 13] variant of Chaum's scheme with NIZK confirmation and disavowal protocols. They claimed that its security is equivalent to the gap Diffie-Hellman (GDH) problem in the random oracle model, where one is allowed to use the decisional Diffie-Hellman (DDH) oracle to solve the CDH problem.

1.2 Our Contributions

In this paper, we first introduce a new kind of adversarial goal called *forge-and-impersonate* in undeniable signature schemes. In the past, the main adversarial goal is *forging* and thus the most desirable security notion is the security against existentially forgery under adaptive chosen message attack [20]. In the new adversary model, the adversary not only attempts to forge but it also attempts to impersonate a legitimate signer. More precisely, an adversary first forges a message-signature pair and next executes a confirmation protocol with a verifier, trying to convince the verifier that the signature is indeed valid. Note that forgeability does not necessarily imply impersonation ability.

We then classify the security of the FDH variant of Chaum's undeniable signature scheme according to three dimensions, the adversarial goals, the attacks and the ZK level of confirmation and disavowal protocols. Finally, we prove the equivalence between each security and some well-known computational problem under various types of confirmation and disavowal protocols as shown in Table 1. However, we cannot solve the three cells marked "?" and it will be a further work to make them clear.

Table 1. The Equivalence

<table>
<tr><th rowspan="2"></th><th colspan="2">forge (F)</th><th colspan="2">forge-and-impersonate (FI)</th></tr>
<tr><th>passive</th><th>active</th><th>passive</th><th>active</th></tr>
<tr><td>ZKIP</td><td colspan="2">CDH
(Theorem 2)</td><td>?</td><td>?</td></tr>
<tr><td>HVZK</td><td>CDH
(Theorem 3)</td><td>?</td><td>DLOG
(Theorem 4)</td><td>$\geq$ one-more DLOG
(Theorem 6)</td></tr>
<tr><td>NIZK</td><td>CDH
(Theorem 1)</td><td>–</td><td>DLOG or break PKS
(Theorem 5)</td><td>–</td></tr>
</table>

*PKS denotes the verifier's public key system

In our result, we also point out that the claim of Okamoto and Pointcheval as mentioned at the end of Section 1.1 is wrong. Following our result from Theorem 1 which is indicated in Table 1, we show that the unforgeability of the FDH variant of Chaum's scheme with NIZK confirmation and disavowal protocols is equivalent to the CDH problem, as opposed to the GDH problem as claimed by them (cf. Claim 1). Further comments on their flaw will be given in Section **??**.

Following is some explanation on Table 1. In the passive attack, the adversary does not interact with the prover. What the adversary does is eavesdropping and she is in possession of transcripts of conversations between the prover and the verifier. In the active attack, the adversary gets to play the role of a cheating verifier, interacting with the prover several times, in an effort to extract some useful information before the forgery or *forge-and-impersonate* attempt. We remark that if the scheme employs the NIZK confirmation and disavowal protocols then it is not necessary to consider the active attack.

Meanwhile, there exists another security notion for undeniable signatures called invisibility which was first introduced by Chaum et al. [12]. This notion is essentially the inability to determine whether a given message-signature pair is valid for a given user. We further prove the invisibility of the FDH variant of Chaum's scheme.

1.3 Organization

The remainder of this paper is organized as follows. In Section 2, we recall the definitions for some computational problems and the definition for undeniable signatures. We also describe the FDH variant of Chaum's scheme and all the confirmation and disavowal protocols associated with it. In Section 3, we present a new adversary model for undeniable signatures. In Section 4, we elaborate on how to classify the security of undeniable signatures. In Section 5, we analyze and discuss the security of the FDH variant of Chaum's scheme under various confirmation and disavowal protocols comprehensively. In Section 6, we point out the flaw in Okamoto and Pointcheval's claim. In Section 7, we discuss the invisibility of the FDH variant of Chaum's scheme under various confirmation and disavowal protocols. Finally, we conclude this paper in Section 8.

2 Preliminaries

2.1 Some Computational Problems

Let G be an Abelian group of prime order q, and let g be a generator of G. We say that (g, g^x, g^r, g^z) is a DH-tuple if $z = xr \bmod q$.

The DDH problem is to decide if (g, g^x, g^r, g^z) is a DH-tuple. The CDH problem is to compute g^{xr} from (g, g^x, g^r). The GDH problem is to solve the CDH problem with the help of a DDH oracle. (Informally, it means that the CDH problem is hard but the DDH problem is easy.) The DLOG problem is to compute x from g^x.

We also briefly define the one-more DLOG problem as follows [3, 4]:
A one-more DLOG adversary is a randomized, polynomial time algorithm M that gets input g and has access to two oracles, namely, a *DLOG oracle* that given $y \in G$ returns $x \in Z_q$ such that $g^x = y$, and a *challenge oracle* that each time it is invoked (it takes no inputs), returns a random challenge point $y \in G$. We say that the adversary M wins if for arbitrary (polynomially bounded) t challenge oracle access, it can find the DLOGs of all the challenges with at most $t - 1$ (strictly less than t) DLOG oracle access.

2.2 Undeniable Signatures

We briefly review the concept of undeniable signatures introduced by Chaum and van Antwerpen [11].

Definition 1. *An undeniable signature scheme consists of the following two polynomial time algorithms and two (possibly non-interactive) protocols between a signer and a verifier.*

- **Key Generation.** *On input the security parameter 1^k, the algorithm produces a pair of matching public and secret keys (pk, sk).*
- **Signing.** *On input a secret key sk and a message m, the algorithm returns a signature σ.*
- **Confirmation Protocol.** *A protocol between a signer and a verifier such that when given a message m, a signature σ and a public key pk, allows the signer to convince the verifier that σ is indeed a valid signature on m for a public key pk, with the knowledge of the secret key sk. If (m, σ) is invalid, then no signer can prove it with non-negligible probability.*
- **Disavowal Protocol.** *A protocol between a signer and a verifier such that when given a message m, a string σ and a public key pk, allows the signer to convince the verifier that σ is an invalid signature on m for a public key pk, with the knowledge of the secret key sk. If (m, σ) is valid, then no signer can prove it with non-negligible probability.*

In the existing literature, the unforgeability for undeniable signatures is similar to the one for ordinary digital signatures, which is the notion of existential unforgeability against adaptive chosen message attack [20]. The only difference is that besides the signing oracle access, the forger of an undeniable signature is also allowed to access to the confirmation/disavowal oracle. The confirmation/disavowal oracle is simulated based on the types of attacks mounted, i.e. passive attack and active attack.

Informally speaking, the forger is given the public key, and after some adaptive signing queries and confirmation/disavowal queries, the forger attempts to produce a valid message-signature pair (m^*, σ^*) such that m^* has never been queried to the signing oracle. We say that the forger succeeds if it outputs such a valid forgery (m^*, σ^*), or it queries a valid (m^*, σ^*) to the confirmation/disavowal oracle such that m^* has never been queried to the signing oracle.

2.3 The FDH Variant of Chaum's Undeniable Signature Scheme

The FDH variant of Chaum's scheme is described as follows. Let G be an Abelian group of prime order q, and let g be a generator of G.

- **Key Generation.** On input the security parameter 1^k, choose $x \in Z_q$ randomly and compute $y = g^x$. Choose a cryptographic hash function $H : \{0,1\}^* \rightarrow G$. Set the public key as (g, y, H) and the secret key as x.
- **Signing.** On input the public key (g, y, H), the secret key x and a message $m \in \{0,1\}^*$, the algorithm returns the signature as $\sigma = H(m)^x$.
- **Confirmation Protocol.** Given a message-signature pair (m, σ), the signer proves that $(g, y, H(m), \sigma)$ is a DH-tuple.
- **Disavowal Protocol.** Given a message-signature pair (m, σ), the signer proves that
  $(g, y, H(m), \sigma)$ is not a DH-tuple.

Remark Chaum's original scheme (which does not employ a cryptographic hash function) is not secure as it is existentially forgeable. Most precisely, it succumbed to the basic multiplicative attack: suppose that an adversary has two message-signature pairs (m_1, σ_1) and (m_2, σ_2), where $\sigma_1 = m_1^x$ and $\sigma_2 = m_2^x$. Then it is obvious that $\sigma_1\sigma_2$ is a signature of m_1m_2.

Confirmation and Disavowal Protocols. There are various confirmation and disavowal protocols associated with Chaum's scheme, each with variable degrees of zero-knowledgeness and efficiency. We make an effort to summarize the various confirmation and disavowal protocols as follows.

Zero-Knowledge Interactive Proof (ZKIP). The first proposal by Chaum and van Antwerpen was not zero-knowledge [11]. In [8], an improved version with *zero-knowledgeness* was proposed. The confirmation protocol is a 4-move ZKIP for language of DH-tuples. For brevity, we describe the complete protocol in Fig. 1-(a).

A somewhat inefficient ZKIP disavowal protocol which requires more than 4-move was also proposed in [8]. A single execution of the protocol is as depicted in Fig. 1-(b). In this figure, com(s) denotes the commitment of s and decom(s) denotes the revealing of s.

Signer		Verifier
		$a, b \xleftarrow{R} Z_q$
	$\xleftarrow{c}$	$c = g^a H(m)^b$
$r \xleftarrow{R} Z_q$		
$z_1 = cg^r$		
$z_2 = z_1^x$	$\xrightarrow{z_1, z_2}$	
	$\xleftarrow{a,b}$	
$c \stackrel{?}{=} g^a H(m)^b$	$\xrightarrow{r}$	
		$z_1 \stackrel{?}{=} g^{a+r} H(m)^b$
		$z_2 \stackrel{?}{=} y^{a+r} \sigma^b$

(a) Confirmation protocol

Signer		Verifier
		$s \xleftarrow{R} \{0, 1, \ldots, k\}$
		$a \xleftarrow{R} Z_q$
		$c = g^a H(m)^s$
	$\xleftarrow{c, c'}$	$c' = y^a \sigma^s$
find s' s.t.		
$(c^x/c') = (H(m)^x/\sigma)^{s'}$	$\xrightarrow{\mathrm{com}(s')}$	
	$\xleftarrow{a}$	
$c \stackrel{?}{=} g^a H(m)^{s'}$	$\xrightarrow{\mathrm{decom}(s')}$	
		$s' \stackrel{?}{=} s$

(b) A single execution of disavowal protocol

Fig. 1. ZKIP protocols

3-Move Honest-Verifier Zero-Knowledge Proof (HVZK). A 3-move honest-verifier zero-knowledge (HVZK) confirmation protocol is depicted in Fig. 2-(a). The corresponding 3-move HVZK disavowal protocol was shown by Camenisch and Shoup recently [7]. We describe the protocol in Fig. 2-(b).

Signer		Verifier
$r \xleftarrow{R} Z_q$		
$z_1 = g^r$		
$z_2 = H(m)^r$	$\xrightarrow{z_1, z_2}$	
	$\xleftarrow{c}$	$c \xleftarrow{R} Z_q$
$d = r + cx \bmod q$	$\xrightarrow{d}$	
		$g^d \stackrel{?}{=} z_1 y^c$
		$H(m)^d \stackrel{?}{=} z_2 \sigma^c$

(a) Confirmation protocol

Signer		Verifier
$s, r, r' \xleftarrow{R} Z_q$		
$w = (H(m)^x/\sigma)^s$		
$z_1 = g^r/y^{r'}$		
$z_2 = H(m)^r/\sigma^{r'}$	$\xrightarrow{w, z_1, z_2}$	$w \stackrel{?}{\neq} 1$
	$\xleftarrow{c}$	$c \xleftarrow{R} Z_q$
$d = r + cxs \bmod q$		
$d' = r' + cs \bmod q$	$\xrightarrow{d, d'}$	
		$g^d/y^{d'} \stackrel{?}{=} z_1$
		$H(m)^d/\sigma^{d'} \stackrel{?}{=} z_2 w^c$

(b) Disavowal protocol

Fig. 2. HVZK protocols

Non-Interactive Zero-Knowledge Proof (NIZK). In general, a 3-move honest-verifier zero-knowledge protocol can be transformed to a more efficient non-interactive zero-knowledge (NIZK) protocol by using the Fiat-Shamir transformation [16, 1], where we need to employ another random oracle H'. However, we cannot use the above solution as a confirmation protocol or a disavowal protocol because such NIZK proof is just an ordinary digital signature.

To overcome this problem, designated-verifier technique was introduced in [21] by Jakobsson et al. In a designated-verifier confirmation proof, the signer proves that "$(g, y, H(m), \sigma)$ is a DH-tuple" or "he knows the verifier's secret key" (the signer knows the former, but not the latter). In other words, the verifier is able to produce such a valid proof himself using his secret key. By using the designated-verifier technique, one can thereby prevent illegal copies of the proof.

Using the technique shown in [14], a designated-verifier proof can be constructed for a public-secret key pair of any well-known public key system. The obtained NIZK proof is zero-knowledge in the random oracle model.

We do not give the concrete NIZK designated-verifier confirmation and disavowal protocols since different protocols are associated with different public key systems used by the verifier.

3 New Adversary Model

In this section, we present a new adversary model for undeniable signatures that incorporates a new adversarial goal called *forge-and-impersonate*. In the past, the main adversarial goal is *forging*, i.e. one considers an undeniable signature scheme to be secure if it is existentially unforgeable against adaptive chosen message attack. In our new proposal, the adversary not only attempts to forge but it also attempts to impersonate a legitimate signer.

It is clear that forgeability does not necessarily imply impersonation ability. Hence the new adversarial goal is stronger. (On the other hand, the latter implies the former because if (m, σ) is invalid, then any signer can convince the verifier with only negligible probability in the confirmation protocol. See Section 2.2.)

Now, we present our proposal and explain what motivates us to consider this new adversarial goal.

As usual, we classify adversaries by their ultimate adversarial goals. Normally, an adversary with the motive to forge a new message-signature pair (m, σ) is given the name *forger*. As mentioned earlier, this is the traditional security notion.

Now, we introduce a new type of adversary. The new adversarial goal is to forge a message-signature pair (m, σ) and further convincing a (honest) verifier that σ is indeed a valid signature on m, by executing the confirmation protocol with the verifier. To avoid confusion, we stick to the following notation. We denote the former type of adversary as *forge* (F) and the latter as *forge-and-impersonate* (FI).

It is pretty hard for this new adversary to gain a success, but let us look at the motivation for the adversary. As noted earlier in the introduction part, the most common application of undeniable signatures is in licensing softwares. If an adversary succeeds in forging a signature (but not in convincing the verifier by executing a confirmation protocol), no doubt it would cause some damage to the legitimate signer (e.g. Microsoft). On the other hand, if an adversary succeeds in forging as well as in impersonating, then it can sell its own softwares by impersonating an agent of Microsoft. In this case, it can actively earn some fast money through its wicked deed. This is the motivation behind the attack.

Intuitively, the security against a FI adversary is equivalent to a problem which is no easier than the problem which is equivalent to the security against a F adversary. We shall exemplify this with some security analyses in the next section.

On the other hand, we also remark that the security against FI does not imply unforgeability from the definitions. From the definition of FI adversary, the adversary forges (m, σ) and succeeds in the confirmation protocol. However, notice that there is a possibility that even if (m, σ) is invalid, the adversary succeeds in the confirmation protocol. Hence, the security against FI adversary does not imply unforgeability. We also note that if we use a ZKIP confirmation protocol, then the security against FI adversary does imply unforgeability, due to the soundness of the ZKIP protocol.

4 Classification of Security Models

4.1 Types of Attacks

We can also classify adversaries by their capabilities or types of attacks. More precisely, there exist two types of attacks, namely, passive attack and active attack. Obviously, passive attack is a weaker attack.

Both the passive and active adversaries have access to the signing oracle as well as the confirmation/disavowal oracle. The signing oracle plays the role similar to those in the ordinary signature scheme. We highlight the difference between a passive attack and an active attack below.

Whenever an adversary submits a confirmation/disavowal query (m, σ), the oracle responds based on whether a passive attack or an active attack is mounted. In a passive attack, the confirmation/disavowal oracle first checks the validity of (m, σ). If it is a valid pair, then the oracle returns "yes" and a transcript of confirmation protocol. Otherwise, the oracle returns "no" and a transcript of disavowal protocol. In an active attack, the confirmation/disavowal oracle first checks the validity of (m, σ). If it is a valid pair, then the oracle returns "yes" and proceeds with the execution of the confirmation protocol with the adversary (acting as a cheating verifier). Otherwise, the oracle returns "no" and executes the disavowal protocol with the adversary accordingly.

4.2 Formal Security Definitions

In this section, we provide the formal security definitions by considering the two adversarial goals, namely *forge* (F) and *forge-and-impersonate* (FI) and the two types of attacks mounted by the adversary.

Definition 2 (Unforgeability). *An undeniable signature scheme is said to be existential unforgeable under adaptive chosen message attack if no probabilistic* *polynomial time (PPT) forger F has a non-negligible advantage in the following game:*

1. *Let pk be the input to F.*
2. *The forger F is permitted to issue a series of queries:*
   - *Signing queries: F submits a message m to the signing oracle and receives a signature σ on m. (We consider adaptive queries here – subsequent queries is made based on previously obtained signatures.)*
   - *Confirmation/disavowal queries: F submits a message-signature pair (m, σ) to the confirmation/disavowal oracle, and the oracle responds based on whether a passive attack or an active attack is mounted.*
     In a passive attack, the confirmation/disavowal oracle first checks the validity of (m, σ). If it is a valid pair, then the oracle returns "yes" and a transcript of confirmation protocol. Otherwise, the oracle returns "no" and a transcript of disavowal protocol.
     In an active attack, the confirmation/disavowal oracle first checks the validity of (m, σ). If it is a valid pair, then the oracle returns "yes" and proceeds with the execution of the confirmation protocol with the forger F (acting as a cheating verifier). Otherwise, the oracle returns "no" and executes the disavowal protocol with F accordingly.
3. *At the end of this attack game, F outputs a message-signature pair (m^*, σ^*).*

The forger F wins the game if F outputs a valid message-signature pair (m^*, σ^*) such that m^* has never been queried to the signing oracle, or it queries a valid (m^*, σ^*) to the confirmation/disavowal oracle such that m^* has never been queried to the signing oracle. F's advantage in this game is defined to be $Adv(F) = \Pr[F wins]$.

Definition 3 (Unforgeability-and-Unimpersonation). *An undeniable signature scheme is said to be secure against forgery and impersonation under adaptive chosen message attack if no PPT adversary A has a non-negligible advantage in the following game:*

1. *Let pk be the input to A.*
2. *The adversary A enters the learning phase where it performs a series of queries: signing queries and confirmation/disavowal queries as in the previous definitions (based on whether a passive attack or an active attack is mounted). At the end of this forgery phase, A outputs a forged message-signature pair (m^*, σ^*) such that m^* has never been queried to the signing oracle.*
   We assume that if A queries (m^*, σ^*) to the confirmation/diavowal oracle such that m^* has never been queried to the signing oracle, and the oracle returns "yes", then A outputs (m^*, σ^*) immediately.
3. *In the impersonation phase, A proceeds to execute the confirmation protocol with a verifier on input (m^*, σ^*), trying to convince the verifier that (m^*, σ^*) is a valid pair.*

The adversary A wins the game if it can convince the verifier that (m^*, σ^*) is a valid message-signature pair. A's advantage in this game is defined to be $Adv(A) = \Pr[A wins]$.

4.3 FI-Security in NIZK

For undeniable signature schemes with designated-verifier NIZK proofs, we have to carefully define the security against FI attack. This is because in such scheme, besides breaking the undeniable signature scheme, an adversary can also impersonate by breaking the public key system of a verifier.

Therefore, we first specify the key generation algorithm of the public key system PKS of the target verifier. We denote the FI attack in this situation with $\mathsf{FI}^{\mathsf{PKS}}$ attack. We then adopt the following adversary model.

1. As usual, after making some oracle queries, the adversary A outputs a forged message-signature pair (m, σ).
2. Now, A is given a public key of a verifier randomly.
3. Next, it outputs a non-interactive non-transferable confirmation transcript corresponding to the given public key.

We say that A succeeds in $\mathsf{FI}^{\mathsf{PKS}}$ attack if the proof is accepted with non-negligible probability, where the probability is taken over the key generation algorithm of PKS as well.

5 The Equivalence

5.1 Our Objective

Following from the previous section, it is thus clear that we need to consider four types of adversaries, namely, the passive F, the active F, the passive FI and the active FI.

There are various confirmation and disavowal protocols associate with the FDH variant of Chaum's scheme, namely, ZKIP, 3-move HVZK and 1-move NIZK.

We intend to explore further on the equivalence between the security of the scheme (with various confirmation and disavowal protocols) and some computational problems, under the various types of adversaries. In other words, our objective is to fill up Table 1.

We remark that if the scheme employs the non-interactive confirmation and disavowal protocols (NIZK), then it is not necessary to consider active attack.

In what follows, a *xxx* scheme denotes the scheme with *xxx* confirmation and disavowal protocols, where *xxx* is ZKIP, HVZK or NIZK.

5.2 On F Attacks

We now show that the passive F attack to the scheme with NIZK protocols is equivalent to the CDH problem. In the NIZK scheme, the public key is (g, y, H, H'), where H' is a hash function which is used for Fiat-Shamir transformation (which transforms a 3-move HVZK protocol to an NIZK proof).

Theorem 1. *The NIZK scheme is secure against passive* F *attack in the random oracle model if and only if the CDH problem is hard.*

Proof. Firstly, we show that if there exists an algorithm M that solves the CDH problem with advantage ϵ_M, then one can construct a forger F that can forge in the universal way with advantage ϵ_F, by running M as a subroutine. The forger F is given the public key (g, y, H, H') where $y = g^x$. For a random message m, F computes $h = H(m)$ and gives the triple (g, y, h) as input to M. When M outputs h^x, F simply outputs the forgery as $(m, \sigma = h^x)$. It is clear that $\epsilon_F = \epsilon_M$. This completes the first half of our proof.

Secondly, we show that if there exists a forger F that manage to forge with advantage ϵ_F, then one can construct an algorithm M that can solve the CDH problem with advantage ϵ_M, by running F as a subroutine. Suppose the input to M is (g, g^x, g^r). M then starts running F by feeding F with the public key $(g, y = g^x, H, H')$ where H and H' are random oracles that will be simulated by M. M also simulates the signing oracle and the confirmation/disavowal oracle itself. Let q_S and q_H be the number of signing queries and H-queries that F issues respectively. We assume that when F requests a signature on a message m_i, it has already made the corresponding H query on m_i.

When F makes a H query for a message m_i, M responds with $h_i = H(m_i) = g^{\alpha_i}$ with probability δ and $h_i = H(m_i) = (g^r)^{\alpha_i}$ with probability $1-\delta$, where α_i is chosen randomly from Z_q and δ is a fixed probability which will be determined later.

When F makes a H'-query for a new *str*, where *str* is the string that F would like to know its H' value, M always responds with a random number. In fact, M assigns some values to $H'(str)$ for some *str* in order to simulate the confirmation/disavowal oracle. When F makes a H'-query for such *str*, M returns $H'(str)$ to F.

Suppose that F makes a signing query for a message m_i. If M has responded with $h_i = g^{\alpha_i}$ to the H query for a message m_i, then M returns $\sigma_i = y^{\alpha_i}$ as the valid signature (since $y^{\alpha_i} = (g^x)^{\alpha_i} = h_i^x = H(m_i)^x$). Otherwise, M aborts and it fails to solve the CDH problem.

Next, we consider the case when F makes a confirmation/disavowal query. Let q_v be the number of queries that F issues to the confirmation/disavowal oracle. For convenience, we consider that the final output of F is the $(q_v + 1)$th query. We say that (m_i, σ'_i) is special if it is a valid message-signature pair queried by F to the confirmation/disavowal oracle such that m_i has never been queried to the signing oracle. M guesses the first special query. More precisely, M guesses the first i such that the ith query (m_i, σ'_i) is special. So, at the beginning, M chooses $Guess \in \{1, 2, \cdots, q_v + 1\}$ randomly. There are two cases to be considered here, namely, $i < Guess$ and $i = Guess$. First suppose that $i < Guess$.

- If F has never made a signing query for m_i, then M returns "no" and the transcript of the disavowal protocol.
- Otherwise, F has already made a signing query for m_i, and M answered with a valid signature σ_i with probability δ (with probability $(1-\delta)$ M aborts).

If $\sigma_i = \sigma'_i$ then M returns "yes" and the transcript of the confirmation protocol. Otherwise, M returns "no" and the transcript of the disavowal protocol.

As mentioned before, M can manipulate the H'-oracle and thus it can generate a transcript of the confirmation or disavowal protocol. (In fact, it is possible that collision occurs for *str*, meaning that *str* is being asked to the H'-oracle by F earlier before M assigns a value to $H'(str)$. However, this probability is negligible and thus it will not affect the overall success probability for M.)

Now suppose that $i = Guess$. Let (m^*, σ^*) be the ith query. If F has queried m^* to the signing oracle, then M aborts. Otherwise, we assume that F has queried the H oracle on m^* and so $m^* = m_j$ for some j. If $h_j = (g^r)^{\alpha_j}$, then we have $\sigma^* = h_j^x = (g^{r\alpha_j})^x$. Consequently, M outputs $g^{xr} = (\sigma^*)^{1/\alpha_j}$ and thus it solves the CDH problem. Otherwise, M aborts and it fails to solve the CDH problem.

To complete the proof, it remains to calculate the probability that M does not abort. M guesses the first special query with probability $1/(q_v + 1)$. The probability that M answers to all the signing queries is δ^{q_S} and M outputs g^{xr} with probability $1 - \delta$. Therefore, the probability that M does not abort during the simulation is $\delta^{q_S}(1-\delta)/(q_v+1)$. This value is maximized at $\delta_{opt} = 1-1/(q_S+1)$. This shows that M's advantage ϵ_M is at least $(1/e(1+q_S))\epsilon_F/(q_v+1)$, where e is the base of the natural logarithm. This is because the value $(1-1/(q_S+1))^{q_S}$ approaches $1/e$ for large q_S. This completes our proof. □

Theorem 2. *The ZKIP scheme is secure against each of passive/active* F *attack in the random oracle model if and only if the CDH problem is hard.*

Proof. The *only if* part is trivial. The *if* part can be shown almost similarly to Theorem 1. Notice that M does not need to simulate the H'-oracle here. The signing oracle and H oracle are simulated similarly (see the proof of Theorem 1).

The confirmation/disavowal oracle is simulated similarly except for the simulation of a transcript. The zero-knowledge property of the protocols assures that M can simulate the confirmation/disavowal oracle. In Appendix A, we will present the concrete simulation of confirmation/disavowal protocol in an active attack, where the verifier is a forger F. M can also simulate the confirmation/disavowal oracle in a passive attack since we have only to consider the honest verifier in the simulation of Appendix A. □

Theorem 3. *The HVZK scheme is secure against passive* F *attack in the random oracle model if and only if the CDH problem is hard.*

Proof. The *only if* part is trivial. The *if* part can be shown almost similarly to Theorem 2 except in the confirmation/disavowal oracle simulation. In Appendix B, we will present the concrete perfect simulation of the transcripts of the confirmation/disavowal protocol. □

5.3 On Passive FI Attacks

Theorem 4. *The passive* FI *attack on the HVZK scheme is equivalent to the DLOG problem in the random oracle model.*

Proof. Firstly, we show that if there exists an algorithm M that solves the DLOG problem, then an adversary A can succeed in FI attack by running M as a subroutine. The adversary A is given the public key (g, y, H) where $y = g^x$. Since A can obtain the secret key x by feeding y to algorithm M, it can succeed in the FI attack. This completes the first half of the proof.

Secondly, let A be a passive FI adversary. We show that one can construct an algorithm M that can solve the DLOG problem by running A as a subroutine. Suppose that the input to M is (g, g^x), M then starts running A by feeding A with the public key $(g, y = g^x, H)$, where H is a random oracle that will be simulated by M. M also simulates the signing oracle and the confirmation/disavowal oracle itself. We assume that when A requests a signature on a message m_i, it has already made the corresponding H query on m_i.

When A makes a H query for a message m_i, M responds with $h_i = g^{\alpha_i}$, where α_i is chosen randomly from Z_q. When A makes a signing query for a message m_i, M returns $\sigma_i = y^{\alpha_i}$ as the valid signature (since $y^{\alpha_i} = (g^x)^{\alpha_i} = h_i^x = H(m_i)^x$).

Next, we consider the case when A makes a confirmation/disavowal query for a message-signature pair (m_i, σ_i'). As in the proof of Theorem 1, let q_v be the number of queries that A issues to the confirmation/disavowal oracle. For convenience, we consider that the forgery of A is the $(q_v + 1)$th query. We say that (m_i, σ_i') is special if it is a valid message-signature pair queried by A to the confirmation/disavowal oracle such that m_i has never been queried to the signing oracle. M guesses the first special query. More precisely, M guesses the first i such that the ith query (m_i, σ_i') is special. So, at the beginning, M chooses $Guess \in \{1, 2, \cdots, q_v + 1\}$ randomly. There are two cases to be considered here, namely, $i < Guess$ and $i = Guess$. First suppose that $i < Guess$.

- If A has never made a signing query for m_i, then M returns "no" and runs the disavowal protocol with A.
- Otherwise, A has already made a signing query for m_i, and M has answered with a valid signature σ_i. If $\sigma_i = \sigma_i'$ then M returns "yes" and runs the confirmation protocol with A. Otherwise, M returns "no" and runs the disavowal protocol with A.

Now suppose that $i = Guess$. Let (m^*, σ^*) be the ith query. If A has queried m^* to the signing oracle, then M aborts. Otherwise, we assume that A has queried the H oracle on m^* and so $m^* = m_j$ for some j. Notice that M can simulate the confirmation/disavowal oracle perfectly since the views of the honest-verifier zero-knowledge protocols are simulatable (see Appendix B).

Next, A proceeds to prove that σ^* is indeed a valid signature corresponding to the message m^* by executing the confirmation protocol with the honest-verifier. Since the confirmation protocol is a proof of knowledge of x, thus M can extract x by using the reset technique [2]. Please refer to Appendix C for the details. $\square$

The following theorem states the security of the scheme against passive FI attack when non-interactive zero-knowledge proofs are used.

Theorem 5. *The passive $\mathsf{FI}^{\mathsf{PKS}}$ attack on the NIZK scheme is equivalent to "solving the DLOG problem or breaking PKS" in the random oracle model. Here, "breaking" PKS means that the adversary obtains the secret key corresponding to the given public key which is chosen randomly in PKS.*

Proof. Consider an algorithm M whose input is $((g, y), Pk)$ where y is a random element of G and Pk is a randomly chosen public key in PKS. If M outputs x such that $y = g^x$ or Sk such that (Pk, Sk) is a public-secret key pair in PKS, then we can say that M succeeds in "solving the DLOG problem or breaking PKS". Clearly, if there exists such algorithm M, then an adversary A can succeed in $\mathsf{FI}^{\mathsf{PKS}}$ attack by running M as a subroutine. Thus the first half of the proof was shown.

Secondly, let A be a passive $\mathsf{FI}^{\mathsf{PKS}}$ adversary. We show that one can construct an algorithm M that can solve the DLOG problem or can break PKS by running A as a subroutine. Suppose that the input to M is $((g, y), Pk)$. At first, M starts running A by feeding A with the public key (g, y, H, H'). We assume that when A requests a signature on a message m_i, it has already made the corresponding H query on m_i.

The simulation of the H oracle and the signing oracle are the same as in the previous proof. The simulation of the H'-oracle is the same as the proof of Theorem 1. The simulation of the confirmation/disavowal oracle is also almost the same as those in the proof of Theorem 4.

Eventually, A needs to run the confirmation protocol with the verifier in order to show that the forgery (m^*, σ^*) is a valid message-signature pair. First, it requests a verifier's public key. M then hands Pk to A. A next generates a non-interactive non-transferable confirmation transcript corresponding to Pk and returns the transcript to M. After that, M resets A. Unlike in the previous proof, M has to rewind A to the point that it has made the H'-query for str where $H'(str)$ is used as a random challenge in the confirmation transcript. Using the same argument of forking lemma [25], if A outputs a NIZK confirmation transcript with non-negligible probability, then rewinding A with a different H' value will result M in getting two confirmation transcripts for a common input (m, σ), with non-negligible probability. From these two transcripts, M can obtain a witness W. At last M outputs W. Remember that the designated-verifier confirmation transcript is a proof of knowledge of x (the signer's secret key) or the verifier's secret key Sk. Therefore, we have $W = x$ or $W = Sk$, that is, M succeeds in solving the DLOG problem or breaking PKS. $\square$

From the above theorem, if the target verifier uses ElGamal cryptosystem, then the passive FI attack on NIZK scheme is equivalent to the DLOG problem. If the target verifier uses RSA cryptosystem, then the passive FI attack on NIZK scheme is equivalent to "solving the DLOG problem or factoring the RSA modulus N" [22].

5.4 On Active FI Attacks

Finally, we consider the last case, the active FI attack. In the active FI attack, the adversary has additional power, i.e. to execute confirmation and disavowal protocols interactively with the signer. M plays the role of the signer in this scenario, interacting with the adversary whenever it receives a confirmation/disavowal query.

Theorem 6. *The HVZK scheme is secure against active* FI *attack in the random oracle model if the one-more DLOG problem is hard.*

Proof. We show that if there exists an adversary A that succeeds in the active FI attack, then there exists an algorithm M that can solve the one-more DLOG problem by running A as a subroutine.

We show how to construct such an algorithm M. Suppose that the input to M is g. At first, M queries its challenge oracle to obtain a random element $y_0 \in G$. M then starts running A with input (g, y_0, H) as A's public key, where H is a random oracle that will be simulated by M. M also simulates the signing oracle and the confirmation/disavowal oracle itself.

When A makes a H query for a message m_i, M responds with $h_i = g^{\alpha_i}$, where α_i is chosen randomly from Z_q. When A makes a signing query for a message m_i, M returns $\sigma_i = y_0^{\alpha_i}$ as the valid signature.

Next, we consider the case when A makes a confirmation/disavowal query. Let q_v be the number of queries that A issues to the confirmation/disavowal oracle as in the proof of Theorem 1. For convenience, we consider that the final output of A is the $(q_v + 1)$th query. We say that (m_i, σ_i') is special if it is a valid message-signature pair queried by A to the confirmation/disavowal oracle such that m_i has never been queried to the signing oracle. M guesses the first special query. More precisely, M guesses the first i such that the ith query (m_i, σ_i') is special. So, at the beginning, M chooses $Guess \in \{1, 2, \cdots, q_v + 1\}$ randomly. There are two cases to be considered here, namely, $i < Guess$ and $i = Guess$. First suppose that $i < Guess$.

- If A has never made a signing query for m_i, then M returns "no" and runs the disavowal protocol with A as shown below.
- Otherwise, A has already made a signing query for m_i, and M answered with a valid signature σ_i. If $\sigma_i = \sigma_i'$ then M returns "yes" and runs the confirmation protocol with A as shown below. Otherwise, M returns "no" and runs the disavowal protocol with A as shown below.

M runs the confirmation protocol with A as follows. M first queries its challenge oracle to obtain y_i. M then gives $(y_i, y_i^{\alpha_i})$ to A as the commitment (the first message) in the confirmation protocol. A returns c_i as the challenge (the second message). Notice that M does not possess the secret key x_0, which is the discrete logarithm of y_0 (i.e. $y_0 = g^{x_0}$). In order to find a response (the third message), M sends $y_i y_0^{c_i}$ to the DLOG oracle and gets the discrete logarithm d_i, which is then returned to A. This is exactly the response that would be returned to A because $d_i = x_i + c_i x_0 \bmod q$, where $y_i = g^{x_i}$ and $y_0 = g^{x_0}$.

M runs the disavowal protocol with A with the help of DLOG oracle in the same way.

Now suppose that $i = Guess$. Let (m^*, σ^*) be the ith query. Eventually, A outputs (m^*, σ^*) and proves the validity of this forgery with the confirmation protocol. If σ^* is a valid signature on m^*, then $\sigma^* = y_0^\alpha$. By using the reset technique [2], M can extract the discrete logarithm of y_0 which is x_0 that satisfies $y_0 = g^{x_0}$.

Suppose that M queries its challenge oracle for $t+1$ times altogether. Finally, for $i = 1, \ldots, t$, M computes the discrete logarithms of all the challenge points as $x_i = d_i - c_i x_0 \bmod q$. □

5.5 Discussion

We have analyzed the security of the FDH variant of Chaum's scheme under various types of confirmation/disavowal protocols using the newly proposed adversary model. Their equivalence with some known computational problems are proven. In conclusion, the results we obtained are as summarized in Table 1, which follows from Theorem 1 to Theorem 6.

The three cells marked "?" are still unsolved at the moment due to the following reasons. In the proofs of Theorem 4 and Theorem 5, M can extract x from $y = g^x$ because the confirmation protocol is a proof of *knowledge* of x, thus there exists a knowledge extractor for x. On the other hand, the perfect zero-knowledge confirmation protocol shown in Fig. 1-(a) is a proof of *language* and not a proof of *knowledge*. Therefore, it is impossible for us to construct such a knowledge extractor. This is the reason why we are unable to prove the equivalence between FI attack and some well-known computational problem by using the same approach. May be there exist some other approaches to prove the equivalence, however we are yet to discover it at the moment.

However, we conjecture that the problem which should be equivalent to the security against passive FI and active FI attacks when ZKIP protocols are employed and the problem which should be equivalent to the security against active F attack when HVZK protocols are employed, should be no easier than the CDH problem. We anticipate the solution in the near future and we encourage more attempts on them.

6 The Flaw in Okamoto and Pointcheval's Claim

Okamoto and Pointcheval [24] made the first attempt to analyze the security of Chaum's scheme by incorporating the full-domain hash (FDH) technique [5, 13]. In other words, they studied the security of the FDH variant of Chaum's scheme in the random oracle model by modeling the hash function H as a random oracle. [4] Okamoto and Pointcheval further claimed that they have solved the more than

[4] Another merit in the FDH variant is that messages may be arbitrary bit strings and do not need to be encoded as group elements as in the original scheme.

10 years open problem, i.e. the security of the FDH variant of Chaum's scheme with NIZK protocols is equivalent to the GDH problem.

Following from Theorem 1, we have disproved their claim, and further proved that the security of the FDH variant of Chaum's scheme with NIZK protocols is in fact equivalent to the CDH problem, a more difficult problem than GDH. In the sequel, we first restate their claim and point out the major flaw in their proof compared to the proof of Theorem 1 which we have shown earlier.

Their claim is as follows.

Claim 1. *[24, Theorem 9] An existential forgery under adaptively chosen message attack for the FDH variant of Chaum's undeniable signature scheme is equivalent to the GDH problem in the random oracle model, where the confirmation and disavowal protocols are NIZK.*

The correctness of the above claim was shown by proving the following [24]:

(1) If there exists an algorithm M that solves the GDH problem, then one can construct a forger F that manage to forge a message-signature pair by running M as its subroutine.
(2) If there exists a forger F that forges a message-signature pair, then one can construct an algorithm M that can solve the GDH problem by running F as its subroutine.

The proof of (1) is wrong. In the proof, the forger F runs the algorithm M as follows. At first, the forger F is given the public key (g, y, H, H') (H' is used to transform HVZK to a non-interactive one). F then chooses m randomly and runs M on input $(g, y, H(m))$. If M submits $(g, y, H(m'), \sigma')$ to the DDH oracle, then F queries to its confirmation/disavowal oracle and returns the answer to M. M finally outputs $H(m)^x$ with non-negligible probability from our assumption. Therefore, F can forge the signature on m as $H(m)^x$ with non-negligible probability.

However, suppose that M submits $(g, y, H(m'), \sigma')$ to the DDH oracle. Then what F can query to its confirmation/disavowal oracle is (m', σ'), but not $(H(m'), \sigma')$. Since F cannot compute m' from $H(m')$, so it cannot query (m', σ'). More precisely, since a prover in the confirmation/disavowal protocol takes only the message m' and its signature σ' as input, simulating the DDH oracle would require to inverse the hash function H, which is obviously impossible! Therefore, F fails to simulate the DDH oracle correctly. This is indeed a critical flaw.

The proof of (2) is redundant. In the proof, the confirmation/disavowal oracle is simulated by the DDH oracle. More precisely, to decide whether the given (m, σ) is a valid pair or not, M asks $(g, y, H(m), \sigma)$ to the DDH oracle, and then simulates the confirmation/disavowal oracle by itself. However, notice that M can decide the validity of (m, σ), since it can simulate the signing oracle by itself (as shown in [24]) and furthermore the signing algorithm is deterministic. Thus the DDH oracle is totally redundant here as it plays no function at all.

7 Invisibility

There exists another security notion for undeniable signatures called invisibility which was first introduced by Chaum et al. [12]. This notion is essentially the inability to determine whether a given message-signature pair is valid for a given signer. In the following, we prove the invisibility of the FDH variant of Chaum's scheme.

We review the formal definition of the notion of invisibility as given below.

Definition 4 (Invisibility). *An undeniable signature scheme is said to have the property of invisibility under adaptive chosen message attack if no PPT distinguisher D has a non-negligible advantage in the following game:*

1. *Let pk be the input to D.*
2. *The distinguisher D is permitted to issue a series of queries: signing queries and confirmation/disavowal queries as in Definition 2 (based on whether a passive attack or an active attack is mounted).*
3. *At some point, D outputs a message m^* which has never been queried to the signing oracle, and requests a challenge signature σ^* on m^*. The challenge signature σ^* is generated based on the outcome of a hidden coin toss b. If $b = 1$, then σ^* is generated as usual using the signing oracle, otherwise σ^* is chosen uniformly at random from the signature space S.*
4. *The distinguisher D performs some signing and confirmation/disavowal queries again with the restriction that no signing query on m^* is allowed, and no confirmation/disavowal query on the challenge message-signature pair (m^*, σ^*) is allowed.*
5. *At the end of this attack game, D outputs a guess b'.*

The distinguisher D wins the game if $b' = b$. D's advantage in this game is defined to be $Adv(D) = |\Pr[b' = b] - \frac{1}{2}|$.

Theorem 7. *The invisibility of the NIZK scheme holds if DDH problem is hard.*

Proof. We show that if there exists a distinguisher D with advantage ϵ_D, then one can construct a DDH distinguisher D' with advantage $\epsilon_{D'}$, by running D as a subroutine. Suppose the input to D' is (g, g^x, g^r, g^t). D' then starts running D by feeding D with the public key $(g, y = g^x, H, H')$ where H and H' are random oracles that will be simulated by D'. D' also simulates the signing oracle and the confirmation/disavowal oracle itself. Let q_S and q_H be the number of signing queries and H queries that D issues respectively. We assume that when D requests a signature on a message m_i, it has already made the corresponding H query on m_i.

When D makes a H query for a message m_i, D' responds with $h_i = H(m_i) = g^{\alpha_i}$ with probability δ and $h_i = H(m_i) = (g^r)^{\alpha_i}$ with probability $1 - \delta$, where α_i is chosen randomly from Z_q and δ is a fixed probability which will be determined later. Suppose that D makes a signing query for a message m_i. If D' has responded with $h_i = g^{\alpha_i}$ to the H query for a message m_i, then D' returns

$\sigma_i = y^{\alpha_i}$ as the valid signature (since $y^{\alpha_i} = (g^x)^{\alpha_i} = h_i^x = H(m_i)^x$). Otherwise, D' aborts and it fails to solve the DDH problem.

Eventually, D outputs a message m^*. We assume that D has queried the H oracle on m^* and so $m^* = m_i$ for some i. If $h_i = (g^r)^{\alpha_i}$, then D' returns the challenge signature $\sigma = (g^t)^{\alpha_i}$. Otherwise, D' aborts and it fails to solve the DDH problem.

Next, D performs some H queries, H' queries, signing queries and confirmation/disavowal queries again with the restriction that no signing queries on m^* is allowed, and no confirmation/disavowal query on the challenge message-signature pair (m^*, σ^*) is allowed.

Finally, D outputs a bit b' which it thinks is equal to the hidden bit b. More precisely, D outputs $b' = 1$ if it finds that (m^*, σ^*) is a valid message-signature pair and it outputs $b' = 0$ if it finds that σ^* is chosen uniformly at random from the signature space S.

Subsequently, D' outputs b'. Note that if (m^*, σ^*) is a valid message-signature pair, then (g, g^x, g^r, g^t) is a DH-tuple. Otherwise (g, g^x, g^r, g^t) is not a DH-tuple. Therefore, if D is an invisibility distinguisher then D' is a DDH distinguisher.

Now, we show how to simulate the confirmation/disavowal oracle. If CDH problem is easy, then DDH problem is easy. Hence D' can solve the DDH problem (without using D) in this case.

Suppose that CDH problem is hard. Then D cannot forge with non-negligible probability because forgery is equivalent to CDH problem from Theorem 1. Now assume that D queries (m_i, σ_i') to the confirmation/disavowal oracle.

- If D has never made a signing query for m_i, then D' returns "no" and a transcript of the disavowal protocol. This is justified because D cannot forge as mentioned above.
- Otherwise, D has already made a signing query for m_i, and D' has answered with a valid signature σ_i. If $\sigma_i = \sigma_i'$ then D' returns "yes" and a transcript of the confirmation protocol. Otherwise, D' returns "no" and a transcript of the disavowal protocol.

D' can generate a transcript of the confirmation/disavowal protocol as shown in the proof of Theorem 1.

To complete the proof, it remains to calculate the probability that D' does not abort. The probability that D' answers to all the signing queries is δ^{q_S} and D' succeeds in distinguishing the DDH problem with probability $1-\delta$. Therefore, the probability that D' does not abort during the simulation is $\delta^{q_S}(1-\delta)$. This value is maximized at $\delta_{opt} = 1 - 1/(q_S + 1)$. This shows that D''s advantage $\epsilon_{D'}$ is at least $(1/e(1 + q_S))\epsilon_D$, where e is the base of the natural logarithm. This is because the value $(1 - 1/(q_S + 1))^{q_S}$ approaches $1/e$ for large q_S. This completes our proof. $\square$

Theorem 8. *The invisibility of the HVZK scheme holds against the passive distinguisher if DDH problem is hard.*

Proof. The proof is almost similar to Theorem 7 also except in the confirmation/disavowal oracle simulation. Refer to Appendix B for the concrete perfect simulation of the transcripts of confirmation/disavowal protocol. □

Theorem 9. *The invisibility of the ZKIP scheme holds against both the passive and active distinguishers if DDH problem is hard.*

Proof. The proof is almost similar to Theorem 7. The zero-knowledge property of the protocols assures that D' can simulate the confirmation/disavowal oracle. Refer to Appendix A for the concrete simulation of confirmation/disavowal oracle in an active attack, where the verifier is D. D' can also simulate the confirmation/disavowal oracle in a passive attack since we have only to consider the honest verifier in the simulation of Appendix A. □

As a result, we can summarize the result for invisibility as follows:

Table 2. Invisibility

	passive D	active D
ZKIP	DDH (Theorem 9)	
HVZK	DDH (Theorem 8)	?
NIZK	DDH (Theorem 7)	–

8 Conclusion

In this paper, we introduced another new adversarial goal called *forge-and-impersonate* in undeniable signature schemes, and this leads to a new adversary model which is slightly stronger than the existing one. We also classified the security of the FDH variant of Chaum's undeniable signature scheme according to three dimensions, the attacks, the adversarial goals and the ZK level of confirmation and disavowal protocols, and then related each security to some well-known computational problem. In addition, we also pointed out the flaw in Okamoto and Pointcheval's claim, i.e. we proved that the unforgeability of the FDH variant of Chaum's scheme with NIZK confirmation and disavowal protocols is equivalent to the CDH problem, as opposed to the GDH problem as claimed by them. Finally, we also discussed the notion of invisibility of the scheme under various confirmation and disavowal protocols.

References

1. M. Abdalla, J. An, M. Bellare and C. Namprempre. From identification to signatures via the Fiat-Shamir transform: minimizing assumptions for security and forward-security. *Advances in Cryptology — EUROCRYPT '02*, LNCS 2332, pp. 418–433, Springer-Verlag, 2002.
2. M. Bellare and A. Palacio. GQ and Schnorr identification schemes: proofs of security against impersonation under active and concurrent attacks. *Advances in Cryptology — CRYPTO '02*, LNCS 2442, pp. 162–177, Springer-Verlag, 2002.
3. M. Bellare, C. Namprempre, D. Pointcheval and M. Semanko. The power of RSA inversion oracles and the security of Chaum's RSA-based blind signature scheme. *Financial Cryptography '01*, LNCS 2339, pp. 319–338, Springer-Verlag, 2002.
4. M. Bellare, C. Namprempre, D. Pointcheval and M. Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, vol. 16, no. 3, pp. 185–215, Springer-Verlag, 2003.
5. M. Bellare and P. Rogaway. The exact security of digital signatures – how to sign with RSA and Rabin. *Advances in Cryptology — EUROCRYPT '96*, LNCS 1070, pp. 399–416, Springer-Verlag, 1996.
6. J. Boyar, D. Chaum, I. Damgård and T. Pedersen. Convertible undeniable signatures. *Advances in Cryptology — CRYPTO '90*, LNCS 537, pp. 189–208, Springer-Verlag, 1990.
7. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. *Advances in Cryptology — CRYPTO '03*, LNCS 2729, pp. 126–144, Springer-Verlag, 2003.
8. D. Chaum. Zero-knowledge undeniable signatures. *Advances in Cryptology — EUROCRYPT '90*, LNCS 473, pp. 458–464, Springer-Verlag, 1990.
9. D. Chaum. Designated confirmer signatures. *Advances in Cryptology — EUROCRYPT '94*, LNCS 950, pp. 86–91, Springer-Verlag, 1995.
10. T. Chaum and T. P. Pedersen. Wallet databases with observers. *Advances in Cryptology — CRYPTO '92*, LNCS 740, pp. 89–105, Springer-Verlag, 1993.
11. D. Chaum and H. van Antwerpen. Undeniable signatures. *Advances in Cryptology — CRYPTO '89*, LNCS 435, pp. 212–216, Springer-Verlag, 1989.
12. D. Chaum, E. van Heijst and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. *Advances in Cryptology — CRYPTO '91*, LNCS 576, pp. 470–484, Springer-Verlag, 1991.
13. J. Coron. On the exact security of full domain hash. *Advances in Cryptology — CRYPTO '00*, LNCS 1880, pp. 229–235, Springer-Verlag, 2000.
14. R. Cramer, I. Damgård and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. *Advances in Cryptology — CRYPTO '94*, LNCS 839, pp. 174–187, Springer-Verlag, 1994.
15. I. Damgård and T. Pedersen. New convertible undeniable signature schemes. *Advances in Cryptology — EUROCRYPT '96*, LNCS 1070, pp. 372–386, Springer-Verlag, 1996.
16. A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. *Advances in Cryptology — CRYPTO '86*, LNCS 263, pp. 186–194, Springer-Verlag, 1987.
17. S. Galbraith and W. Mao. Invisibility and anonymity of undeniable and confirmer signatures. *Topics in Cryptology — CT-RSA '03*, LNCS 2612, pp. 80–97, Springer Verlag, 2003.

18. S. Galbraith, W. Mao and K. G. Paterson. RSA-based undeniable signatures for general moduli. *Topics in Cryptology — CT-RSA '02*, LNCS 2271, pp. 200–217, Springer Verlag, 2002.
19. R. Gennaro, H. Krawczyk and T. Rabin. RSA-based undeniable signatures. *Advances in Cryptology — CRYPTO '97*, LNCS 1294, pp. 132–149, Springer-Verlag, 1997.
20. S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptative chosen-message attacks. *SIAM Journal of Computing*, vol. 17, no. 2, pp. 281–308, 1988.
21. M. Jakobsson, K. Sako and R. Impagliazzo. Designated verifier proofs and their applications. *Advances in Cryptology — EUROCRYPT '96*, LNCS 1070, pp. 143–154, Springer-Verlag, 1996.
22. A. May. Computing the RSA secret key is deterministic polynomial time equivalent to factoring. *Advances in Cryptology — CRYPTO '04*, LNCS 3152, pp. 213–219, Springer-Verlag, 2004.
23. M. Michels and M. Stadler. Efficient convertible undeniable signature schemes. *Selected Areas in Cryptography — SAC '97*, pp. 231–244, Springer-Verlag, 1997.
24. T. Okamoto and D. Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. *Public Key Cryptography — PKC '01*, LNCS 1992, pp. 104–118, Springer-Verlag, 2001.
25. D. Pointcheval and J. Stern. Security proofs for signature schemes. *Advances in Cryptology — EUROCRYPT '96*, LNCS 1070, pp. 387–398, Springer-Verlag, 1996.

A Simulation of ZKIP Confirmation/Disavowal protocol

If a protocol has zero-knowledge property, it is possible for the simulator to convince the verifier that the given input is valid without knowing any special information (such as the signing key) nor having infinitely computational power provided that it can reset the verifier. The above argument holds for any verifier.

In this section, we give a concrete simulator for the ZKIP confirmation and disavowal protocols respectively.

Simulator for the Confirmation Protocol on (m, σ)*.*

1. Receive c from the verifier.
2. Compute $z_1 = cg^r$ and $z_2 = y^r$ for a random number r, and send z_1, z_2 to the verifier.
3. Receive a and b from the verifier.
4. If $c \neq g^a H(m)^b$ then abort. Otherwise, reset the verifier.
5. Receive c again.
6. Compute $z'_1 = g^{a+r} H(m)^b$ and $z'_2 = y^{a+r} \sigma^b$ for a random number r, and send z'_1, z'_2 to the verifier.
7. Receive a' and b' again.
8. Send r to the verifier.

Note that the c received in step 5 is the same as the c received in step 1. Also, it is easy to see that $z'_1 = z_1$ and $z'_2 = z_2$. Hence $a' = a$ and $b' = b$.

Simulator for the Disavowal Protocol on (m,σ). Repeat the following steps a predetermined times.

1. Receive c, c' from the verifier.
2. Choose s' randomly from $\{0,1,\ldots,k\}$ and send a commitment of s' to the verifier.
3. Receive a from the verifier.
4. Find $s"$ such that $c = g^a H(m)^{s"}$.
5. If $s" = s'$, then decommit s'. Otherwise, reset the verifier and goto step 1.

In step 5, $s" = s'$ holds with probability $1/(k+1)$ because s' is randomly chosen from $\{0,1,\ldots,k\}$. Hence the simulator resets the verifier $k+1$ times in average.

B Simulation of Transcripts of HVZK Confirmation/Disavowal Protocol

First, we describe how to simulate a transcript of HVZK confirmation protocol for a valid message-signature pair (m,σ). We assume that m has already been asked to the random oracle H and $H(m) = g^\alpha$ for a random element α. (In the rest of this paper, we assume the same situation.) Since (m,σ) is a valid pair, $\sigma = H(m)^x = y^\alpha$.

Recall that as depicted in Fig. 2-(a), a real transcript is (z_1, z_2, c, d) such that c is a random number and

$$\begin{aligned} z_1 &= g^r \\ z_2 &= H(m)^r \\ d &= r + cx \bmod q \end{aligned}$$

for a random element $r \in Z_q$. The check equations are as follows:

$$\begin{aligned} z_1 &= g^d / y^c \\ z_2 &= H(m)^d / \sigma^c. \end{aligned}$$

To compute a transcript, we first choose c and d randomly from Z_q, and compute z_1 and z_2 from the above equations. Then the distribution of (z_1, z_2, c, d) is equivalent to that of a real transcript. Remember that it is enough to simulate a transcript between a signer and an *honest* verifier, that is, c is always a random value.

Next, we explain how to simulate a transcript of HVZK disavowal protocol for an invalid message-signature pair (m,σ). Please refer to Fig. 2-(b) for the real transcript of HVZK disavowal protocol. Since (m,σ) is *not* a valid pair, $\sigma \neq y^\alpha$.

To make a transcript which is indistinguishable from a real transcript, we first choose $s \in Z_q$ randomly and compute $w = (y^\alpha/\sigma)^s$. (Here, y^α is a valid signature of m.) Next, choose $c, d, d' \in Z_q$ randomly, and compute

$$\begin{aligned} z_1 &= g^d / y^{d'}, \\ z_2 &= H(m)^d / (w^c \sigma^{d'}). \end{aligned}$$

Then (w, z_1, z_2, c, d, d') is a transcript for the disavowal protocol.

C Knowledge Extractor of HVZK Confirmation Protocol

First, the knowledge extractor K runs the signer (or the malicious prover) and gets a view of the protocol (z_1, z_2, c, d). If they satisfy $g^d = z_1 y^c$ and $H(m)^d = z_2 \sigma^c$, then it resets the signer and runs it once more with a fresh c'. Otherwise, K aborts. Let the second view be (z_1, z_2, c', d'). If they satisfy $g^{d'} = z_1 y^{c'}$ and $H(m)^{d'} = z_2 \sigma^{c'}$, moreover, if $c \neq c'$, then outputs $x = (d - d')/(c - c') \bmod q$, otherwise aborts.

If K outputs x, then x is the discrete logarithm of y. The probability that K outputs x is $(\epsilon - 1/q)^2$, where ϵ is the success probability of the prover [2].