Secret Signatures: How to Achieve Business Privacy Efficiently?

Byoungcheon Lee[1]*, Kim-Kwang Raymond Choo[2]**, Jeongmo Yang[1], and Seungjae Yoo[1]

[1] Department of Information Security, Joongbu University
101 Daehak-Ro, Chubu-Myeon, Geumsan-Gun, Chungnam, 312-702, Korea
{sultan,jmyang,sjyoog}@joongbu.ac.kr
[2] Australian Institute of Criminology
GPO Box 2944, Canberra ACT 2601, Australia
raymond.choo@aic.gov.au

Abstract. Digital signatures provide authentication and non-repudiation in a public way in the sense that anyone can verify the validity of a digital signature using the corresponding public key. In this paper, we consider the issues of (1) signature privacy and (2) the corresponding public provability of signature. We propose a new digital signature variant, *secret signature*, which provides authentication and non-repudiation to the designated receiver only. If required, the correctness of the secret signature can be proven to the public either by the signer or the receiver. We conclude with a discussion to demonstrate the usefulness of the proposed cryptographic primitive (e.g., achieving signature privacy in an efficient manner).
Keywords: secret signature, signature privacy, public provability, key agreement, anonymity, public auction.

1 Introduction

Digital signature, first proposed by Diffie and Hellman in 1976 [11], is an electronic version of handwritten signatures for digital documents. A digital signature on some message, m, is generated by a signer, A, using a secret signing key, sk_A. The correctness of the generated signature is verified using the corresponding public key, pk_A. It provides authentication and non-repudiation in a public way, in the sense that anyone can verify the validity of the digital signature, since pk_A is public information.

* This work was supported by Korea Research Foundation Grant funded by Korea Government (MOEHRD, Basic Research Promotion Fund), grant No. KRF-2005-003-D00375.

** The views and opinions expressed in this paper are those of the author and do not reflect those of the Australian Government or the Australian Institute of Criminology. This research was not undertaken as part of the author's work at the Australian Institute of Criminology.

Signature Privacy. In this paper, we consider a business scenario where both the sender (signer) and the receiver (verifier) wish to keep their exchanged signatures private – we term this *signature privacy*. Such a (signature privacy) property does not appear to be easily achieved using general digital signature schemes. A 'straightforward' approach would be to encrypt a digital signature with the receiver's public key (or with an agreed key) so that only the legitimate receiver can decrypt and retrieve the original signature. This is the so-called *sign-then-encrypt* approach, which is widely adopted in the real world. In order to implement the signature and encryption operations in a more efficient manner, signcryption [26] was proposed in 1997 by Zheng. Alternative solutions to achieve signature privacy include using special signature schemes that limit the verifiability of the signature only to a designated entity (e.g., the designated verifier signature (DVS) [6, 15] and the limited verifier signature (LVS) [1, 9]).

Public Provability of Signature. Assume that we use a signature scheme designed to provide *signature privacy*. In the event that a dispute arises between a signer and a receiver during a business transaction, any chosen third party (e.g., a judge or the public) should be able to prove the correctness of the digital signature. We term such a property the *public provability* of a signature. This property can be easily achieved using either general digital signature schemes (i.e., by verifying the signature using the signer's public key) or the sign-then-encrypt schemes (i.e., by decrypting the encrypted signature and verifying the retrieved signature in a standard way). In the latter sign-then-encrypt approach, proving the correctness of decryption is a computationally expensive operation (e.g., include zero-knowledge proofs).

If we use signature schemes designed to provide signature privacy, then the public provability of generated signatures becomes an important requirement to ensure the fairness of business transactions. Although signcryption schemes appear to provide such a public provability feature, the original authors have not specified (this feature) explicitly. DVS and LVS schemes. DVS and LVS schemes, on the other hand, are unable to provide the public provability feature, since the designated verifier cannot transfer his conviction to others as the designated verifier is able to open the signature in any way of his choice using the knowledge of his/her private key.

Our approach. In this paper, we introduce a new digital signature variant, *secret signature (SS)*, designed to provide signature privacy. Advantages of our proposed SS scheme include providing all the following properties efficiently:

1. Authenticity and non-repudiation;
2. Signature privacy; and
3. Public provability of signature.

More specifically, in our proposed SS scheme:

- A signer, A, computes a signature using A's private key together with the public key of a receiver, B.
- B can then verify the signature using B's private key and A's public key.

– Given a (signed) message, no one other than the designated receiver can identify the message's authorship (e.g., what message is signed by whom and addressed to whom).
– If required, either A or B can provide a proof of validity of the signature. Given the proof of validity, any third party will be able to verify the validity of the associated signature.

To obtain the above-mentioned functionalities, we combine a secure signature scheme with a non-interactive one-way key agreement scheme between a signer and a receiver. In other words, our proposed SS can be viewed as a signature on a message and a secret agreed key. The designated receiver can recover the secret agreed key and verify the signature, but no third party can identify what message is signed by whom and addressed to whom.

Secret signature is useful in many real-world applications where

1. the privacy of the generated signature needs to be maintained, but the authorship of the signature can be publicly proven at a later stage,
2. message confidentiality is not required, and
3. efficiency is critical.

A main distinction between our proposed SS scheme and general signcryption schemes[3] is that SS scheme provides signature privacy without using encryption. Thus it is more efficient than signcryption in many applications where message confidentiality is not required. We describe some application examples in Section 7.

Outline. We define the proposed SS scheme and provide security definitions in Section 2. In the following section, a concrete construction example of SS in discrete logarithm (DL)-based cryptosystems and the security proofs in the random oracle model are presented. We then present a brief discussion on how to prove the validity of SS in Section 4. We compare the features of SS with previously published signature privacy-related schemes in Section 5 and compare the efficiency of SS with signcryption in Section 6. Several possible applications are presented in Section 7. Section 8 concludes this paper.

2 Definitions

2.1 Definition of Secret Signature Scheme

There are two entities in the SS scheme, namely, a signer (sender), A, and a verifier (receiver), B. The formal definition of SS scheme is as follows.

Definition 1 (Secret Signature Scheme) *A secret signature (SS) scheme consists of the following six algorithms.*

[3] In this paper, we do not consider various features provided by different variants of the signcryption scheme (e.g., [10] and [17]).

1. **Setup :** $\mathsf{params} \leftarrow SS.SetUp(1^k)$**.**
   A probabilistic algorithm, $SS.SetUp$, which takes a security parameter, k, as input and outputs the public parameters, params*.*
2. **Key Generation :** $(pk, sk) \leftarrow SS.KeyGen(\mathsf{params})$**.**
   A probabilistic algorithm, $SS.KeyGen$, which takes the public parameters, params*, as input and outputs a pair (pk, sk) of matching public and private keys. For example, the public/private key pairs of the signer and receiver, (pk_S, sk_S) and (pk_R, sk_R), are generated using $SS.KeyGen$.*
3. **Signing :** $(V, \mathsf{seed}) \leftarrow SS.Sign(\mathsf{params}, m, sk_S, pk_R)$**.**
   A probabilistic algorithm, $SS.Sign$, run by the signer, which takes as input the public parameters, params*, a plaintext message, $m \in \{0,1\}^*$, signer's private key, sk_S, and receiver's public key, pk_R; and outputs a secret signature, V and the random* seed *which was used to compute the signature. Signer has to keep* seed *secretly by himself.*
4. **Verification :** $\mathsf{result} \leftarrow SS.Verify(\mathsf{params}, V, m, pk_S, sk_R)$**.**
   A deterministic algorithm, $SS.Verify$, run by the receiver, which takes as input the public parameters, params*, a secret signature, V, a plaintext message, m, the signer's public key, pk_S, and receiver's private key, sk_R, and outputs* result*. If V is a valid secret signature, then* result $= valid$*, otherwise,* result $= invalid$*. If correct signature $V = SS.Sign(\mathsf{params}, m, sk_S, pk_R)$ is tested, the verification result $SS.Verify(\mathsf{params}, V, m, pk_S, sk_R) \mapsto \mathsf{result}$ should be valid.*
5. **Public Proving.**
   A probabilistic algorithm that is run by either the signer or the receiver to prove the validity of the secret signature to public.
   Run by the signer : $\mathsf{proof_S} \leftarrow SS.Proof.Signer(\mathsf{params}, V, \mathsf{seed})$**.**
   $SS.Proof.Signer$ which takes as input required parameters, params*, the signer's random* seed *used to compute the secret signature, and the secret signature, V, and outputs a proof,* $\mathsf{proof_S}$*.*
   Run by the receiver : $\mathsf{proof_R} \leftarrow SS.Proof.Receiver(\mathsf{params}, V, sk_R)$**.**
   $SS.Proof.Receiver$ which takes as input required parameters, params*, the secret signature, V, and the receiver's private key, sk_R, and outputs a proof,* $\mathsf{proof_R}$*.*
6. **Public Verification :** $\mathsf{result} \leftarrow SS.PubVerify(\mathsf{params}, m, V, pk_S, pk_R, \mathsf{proof})$**.**
   A deterministic algorithm, $SS.PubVerify$, which takes as input the public parameters, params*, a message m, a secret signature, V, the public keys of the signer and the intended receiver, and the validity proof* proof *(either* $\mathsf{proof_S}$ *or* $\mathsf{proof_R}$ *), and outputs a verification result,* result *(either valid or invalid).*

2.2 Security Definitions

Informally we consider the following security requirements for the proposed SS scheme described in Definition 1.

Correctness. If a secret signature is generated by following the protocol correctly, then the result of the verification always return *valid*.

Unforgeability. Anyone except the signer can have a non-negligible advantage in forging a secret signature.

Non-Repudiation. The signer is unable to repudiate the generation of a secret signature that the signer has previously generated. If unforgeability is provided, then non-repudiation is obtained consequently.

Signature Privacy. The secret signature generated by the signer is verifiable only by the designated receiver. No other entity except the signer and the receiver is able to have a non-negligible advantage in distinguishing the secret signature. Signature privacy is defined in terms of invisibility.

Public Provability. The validity of the signature can be proven to public by the signer or the verifier, if the need arises.

To define the unforgeability and non-repudiation more formally, we recall the widely accepted security notions on digital signatures, unforgeability against chosen-message attacks. In order to provide non-repudiation, we would like to prevent the forgery of A's signature without knowledge of A's signing key, except with negligible probability. As shown in the seminal paper of Diffie and Hellman [11], the security of such a scheme in the public key setting typically depends on the existence of a one-way function. A formalized and widely accepted security notion for digital signature was introduced by Goldwasser, Micali, and Rivest [14], which they term *existential unforgeability under adaptive chosen-message attack* (EF-ACMA).

However, in our proposed SS scheme, there are two inputs: the message to be signed and the intended recipient's public key. Hence, we extend the standard security definition to the *existential unforgeability under the adaptive chosen-message chosen-receiver attack* (EF-ACMCRA) in which the attacker is allowed to query secret signatures to the signing oracle for any chosen message and receiver's public key adaptively. An unforgeability of secret signature can be defined in terms of the following unforgeability game.

Game Unforgeability: Let $\mathcal{F}$ be a forger and k be a security parameter.

1. (Initialization) First, $\mathsf{params} \leftarrow SS.SetUp(1^k)$ is executed and the signer's key pair $(pk_S, sk_S) \leftarrow SS.KeyGen(\mathsf{params})$ is computed. pk_S is given to $\mathcal{F}$.
2. (Training) $\mathcal{F}$ is allowed to ask a series of $SS.Sign$ queries for any combination of message m and receiver's public key, pk_R, chosen by $\mathcal{F}$ to the signing oracle. To do this, $\mathcal{F}$ computes $(pk_R, sk_R) \leftarrow SS.KeyGen(\mathsf{params})$, and asks secret signature to the signing oracle by sending (m, pk_R, sk_R). Then the signing oracle provides valid secret signatures V.
3. (Output) $\mathcal{F}$ outputs a pair (m', pk'_R, sk'_R, V') as a forgery of a secret signature on message m' from the signer S to a receiver R'.

$\mathcal{F}$ wins the game if $\mathsf{valid} \leftarrow SS.Verify(\mathsf{params}, V', m', pk_S, sk'_R)$ and the tuple (m', pk'_R, sk'_R, V') has never been queried to $SS.Sign$. In this definition of unforgeability we assume that receiver's key pair is known to $\mathcal{F}$ and the signing oracle. Without the knowledge of receiver's private key the signing oracle cannot simulate secret signature and $\mathcal{F}$ cannot verify the validity of received secret

signature. Since the main concern of the unforgeability game is the unforgeability of signer's signature, this assumption is reasonable.

Definition 2 ***(Unforgeability)*** *A secret signature scheme is said to be secure in the sense of existential unforgeability under the adaptive chosen-message chosen-receiver attack (EF-ACMCRA), if no probabilistic, polynomial-time (PPT) forger, $\mathcal{F}$, can have a non-negligible advantage in Game Unforgeability.*

Signature privacy requires that a given secret signature is a private information between the signer and the receiver. Any other entity cannot distinguish a secret signature from a random string in a signature space. Signature privacy can be defined in terms of the following invisibility game.

Game Invisibility: Let $\mathcal{D}$ be a distinguisher. First, $\mathsf{params} \leftarrow SS.SetUp(1^k)$ is executed and the signer's key pair $(pk_S, sk_S) \leftarrow SS.KeyGen(\mathsf{params})$ is computed. pk_S is given to $\mathcal{D}$. Let $(pk_R, sk_R) \leftarrow SS.KeyGen(\mathsf{params})$ be the receiver's key pair. pk_R is given to $\mathcal{D}$. At some point $\mathcal{D}$ outputs a message m' and requests for a challenge secret signature V' to the challenger $\mathcal{C}$. The challenge V' is generated by $\mathcal{C}$ based on the outcome of a hidden coin toss b. If $b = 1$, V' is generated by running $SS.Sign$. If $b = 0$, V' is chosen randomly in the signature space. At the end of the game, $\mathcal{D}$ outputs a guess b'. $\mathcal{D}$ wins if $b = b'$ and the tuple (m', V') has never been queried to $SS.Sign$.

In this invisibility game, receiver's private key sk_R is hidden from $\mathcal{D}$, since $\mathcal{D}$ is not the designated receiver. The designated receiver can distinguish the secret signature using his private key.

Definition 3 ***(Invisibility)*** *A secret signature scheme is said to provide invisibility, if no probabilistic, polynomial-time distinguisher, $\mathcal{D}$, can have a non-negligible advantage in Game Invisibility.*

2.3 General Implementation

The underlying intuition behind the general implementation of the proposed SS schemes is to combine a secure signature scheme and a non-interactive one-way key agreement. We denote the signer as A and the intended receiver as B.

Key agreement. Assume that A wants to send a secret signature for a message m to B. Using a non-interactive one-way key agreement scheme A generates an agreed secret (session) key, K, using B's public key, pk_B. For example, in DL-based cryptosystems, A chooses a random seed r_A and computes an agreed key $K = pk_B^{r_A} = g^{x_B r_A} = (g^{r_A})^{x_B}$.

Signing. A generates a signature, V, by signing $m||K$ with A's signing key, sk_A. We can interpret V as a secret signature for the message m that is privately shared between A and B, since no one else should be able to verify V without knowledge of K.

Verification. B is able to compute the shared key using his private key and, hence, verify the secret signature. Any entity other than the signer and intended receiver cannot compute K, thus cannot determine the validity of the signature, even cannot tell what message was signed by whom to which receiver.

Note that signatures on the same message generated by the same signer for the same receiver will differ from one session to another due to the additional session key component in the generation of the secret signature. Also the session key K provides a binding between the signature and the recipient, thus the same signature cannot be used for other recipient.

We remark that we mainly focus on the signature privacy, rather than the message confidentiality. Depending on the requirement of the applications, the nature in which the actual message is exchanged can be sent in clear, sent in ciphertext, or does not need to be sent (implicitly known to the receiver).

3 DL-based Implementation of Secret Signature Scheme

The proposed SS scheme can be implemented using different public key cryptosystems (e.g., identity-based cryptosystems). In this section, we present an implementation example of the SS scheme in the discrete logarithm-based setting.

1: Setup
We assume common system parameters (p, q, g) where p and q are large primes satisfying $q|p-1$ and g is an element of order q in $\mathbb{Z}_q^*$. We then require a secure cryptographic hash function, $\mathcal{H} : \{0,1\} \mapsto \mathbb{Z}_q$, which we will model as a random oracle [4]. For readability, we will omit the modulo p in our subsequent equations, if it is clear. Let $\in_R$ denote uniform random selection.

2: Key Generation
A signer A has a long-term certified key pair (x_A, y_A), where $x_A \in_R \mathbb{Z}_q^*$ and $y_A = g^{x_A}$. A receiver B has a long-term certified key pair (x_B, y_B), where $x_B \in_R \mathbb{Z}_q^*$ and $y_B = g^{x_B}$.

3: Signing
Let m denote the message to be signed. The signer, A, selects a random seeds $r_A \in_R \mathbb{Z}_q^*$. Using r_A, A now computes $U = g^{r_A}$ and the key to be shared with the verifier, $W = y_B^{r_A}$. A computes $V = r_A + x_A\mathcal{H}(m, U, W)$. A sends the secret signature, $\langle m, U, V\rangle$, to the intended receiver, B.

4: Verification
The receiver, B, uses his private key, x_B, to compute the exchanged key chosen by the signer, $W = U^{x_B}$. B then verifies V by $g^V \stackrel{?}{=} U \cdot y_A^{\mathcal{H}(m,U,W)}$. If V verifies correctly, then B knows that the message is indeed signed by A.

5: Public Proving
The validity of secret signature $\langle m, U, V\rangle$ is proven to public either by the signer or the receiver. In this stage we consider the following two cases according to the information revealed.

(1) Message proving. If the receiver information needs not be exposed, just reveal W. With the additional information, W, anyone can verify that (U, V) is a correct signature of the signer A for the message m and W. If only message proving is required, it is very efficient.

(2) Receiver proving. If the receiver information needs to be proven, reveal W and prove its correctness with respect to the receiver's public key. Its validity can be proven by the signer or the receiver either non-anonymously (using the general proof) or anonymously (using the anonymous proof) which will be described in Section 4.

6: Public Verification

Given W which is proven to be correct, anyone can verify the validity of the secret signature by checking $g^V \stackrel{?}{=} U \cdot y_A^{\mathcal{H}(m,U,W)}$.

Signer A		Receiver B
	2. Key Generation	
$x_A \in_R \mathbb{Z}_q^*,\ y_A = g^{x_A}$		$x_B \in_R \mathbb{Z}_q^*,\ y_B = g^{x_B}$
	3. Signing	
$r_A \in_R \mathbb{Z}_q^*$ $U = g^{r_A}; W = y_B^{r_A}$ $V = r_A + x_A \mathcal{H}(m, U, W)$	$\xrightarrow{\langle m, U, V \rangle}$	
	4. Verification	
		$W = U^{x_B}$ $g^V \stackrel{?}{=} U \cdot y_A^{\mathcal{H}(m,U,W)}$
	5. Public Proving	
(1) Expose W		(1) Expose $W = U^{x_B}$
(2) Prove the validity of W		(2) Prove the validity of W
	6. Public Verification	
	$g^V \stackrel{?}{=} U \cdot y_A^{\mathcal{H}(m,U,W)}$	

Fig. 1. DL-based implementation

We prove the security of our scheme assuming the intractability of the discrete log problem and also in the random oracle model (ROM).

Theorem 1 *The proposed SS scheme is EF-ACMCRA secure (in the sense of Definition 2) in the random oracle model under the assumption that the discrete logarithm problem is intractable.*

Proof Sketch. Since the proposed SS scheme is a ElGamal family signature scheme, Forking lemma [21, 20] can be applied. We assume that there exists a forger $\mathcal{F}$ (described in Definition 2) that can forge a secret signature in time $t(k)$ with a non-negligible advantage $\epsilon(k)$. In the training stage $\mathcal{F}$ can ask signing queries for any combination of message and receiver pair to the signing oracle and receive correct secret signatures from the signing oracle. The challenger $\mathcal{C}$ controls all communication of $\mathcal{F}$ and simulate the signing queries.

The signing algorithm uses a hash function which is modeled by an random oracle under the random oracle model. For each signing query (M, y_R, x_R) given by $\mathcal{F}$, $\mathcal{C}$ picks random integers $a, b \in_R Z_q^*$ and computes

$$U \leftarrow g^a y_A^b, \quad W \leftarrow U^{x_R}, \quad h \leftarrow -b, \quad V = a.$$

$\mathcal{C}$ gives h as a random oracle answer to the $\mathcal{H}(m, U, W)$ query and (U, V) as the signature for (M, y_R, x_R) signing query. Then this simulated signatures can pass $\mathcal{F}$'s signature verification and are indistinguishable from the real signatures.

The remaining proof is the same as the case for the original Schnorr signature. The discrete logarithm problem can be solved in time $t'(k)$ with advantage $\epsilon'(k)$ where

$$t'(k) \approx \{2(t(k) + q_H\tau) + O_B(q_S k^3)\}/\epsilon(k), \quad \epsilon'(k) \approx q_H^{-0.5}$$

where q_S and q_H are the numbers of signing queries and hash oracle queries, respectively, and τ is time for answering a hash query. If a successful forking is found, signer's private key x_A can be computed, which contradicts the discrete logarithm assumption. □

Theorem 2 *The proposed SS scheme provides signature privacy (invisibility) in the sense of Definitions 3 under the random oracle model, if the decisional Diffie-Hellman (DDH) problem is intractable.*

The proof for Theorem 2 generally follows that of Galbraith and Mao [13]. We assumes that there exist an adversary $\mathcal{D}$, who can gain a non-negligible advantage in distinguishing the signatures in the game outlined in Definition 3. We now construct another algorithm, $\mathcal{D}_{DDH}$, to break the decisional Diffie–Hellman (DDH) problem using $\mathcal{D}$.

4 Proving the Validity of Secret Signature

In the public proving stage, the signer or the receiver prove the validity of secret signature to a judge or public. Once the proof is given, anyone can verify the validity of secret signature and non-repudiation is provided. Here we consider the following two cases.

General Proof. In this protocol, the identity of the entity (signer or receiver) who proves the validity of the SS is revealed, since the signer's proof and the receiver's proof are distinguishable.

Anonymous Proof. As the name suggests, the identity of the entity who proves the validity of the SS is not revealed, since the signer's proof and the receiver's proof are indistinguishable. However, this proof is computationally more expensive than that of the general proof.

4.1 General Proof Protocol

In this protocol, either the signer, A, or the receiver, B, reveals the shared key W and proves its validity using the proof outlined in Appendix A.

- The signer A proves $ZKP(r_A) : (\log_g U = \log_{y_B} W = r_A)$ using his knowledge of r_A.
- The receiver B proves $ZKP(x_B) : (\log_g y_B = \log_U W = x_B)$ using his knowledge of x_B.

It is easy to see that the proofs initiated by the signer and the receiver are distinguishable, hence the identity of the entity who had exposed the secret signature is revealed.

4.2 Anonymous Proof Protocol

There might exist situations where we are unable to reveal the identity of the entity who exposed the secret signature, perhaps, due to privacy or legal restrictions. In such cases, we cannot employ the general proof protocol presented above. Here we show how the signer or the receiver can prove the validity of secret signature anonymously without revealing their identity.

Note that the tuple $\langle g, U, y_B, W\rangle$ has the special relations depicted in Figure 2. The signer knows $r_A (= \log_g U = \log_{y_B} W)$ and the receiver knows $x_B (= \log_g y_B = \log_U W)$.

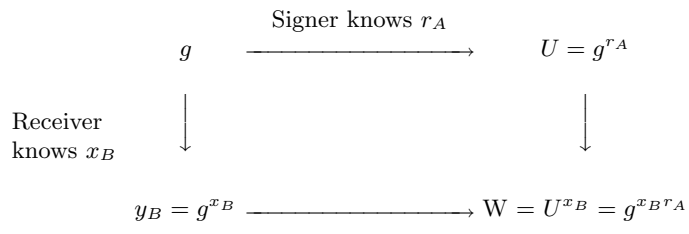

Fig. 2. Special relations of the tuple $\langle g, U, y_B, W\rangle$

The anonymous proof protocol can be initiated either by A or by B. They expose the shared key $W = y_B^{r_A} = U^{x_B} = g^{r_A x_B}$ and demonstrate their knowledge of the corresponding secret information as follows.

$$ZKP(r_A \vee x_B) : (\log_g U = \log_{y_B} W = r_A) \vee (\log_g y_B = \log_U W = x_B).$$

It is a OR combination of two zero-knowledge proofs of the equality of two discrete logarithms described in Appendix B. A or B is able to prove the validity of W by using their knowledge of r_A or x_B.

Although these two proofs by the signer and the receiver are computed differently, any public verifier is unable to distinguish whether the proof is provided by the signer or the receiver. If one of the party opens the secret signature anonymously, the other partner know that no one other than the partnering entity has opened the secret signature. However, the party is unable to prove that the other partnering entity has opened the secret signature. From the public's perspective, the identity remains anonymous.

5 Comparison of Features

Several signature variants found in the literature also provide signature-privacy-related functionalities. We now compare these schemes with our proposed SS scheme.

Sign-then-encrypt approach: Although this approach provides signature privacy property, it has the following disadvantage. Once the receiver decrypt the encrypted signature and obtain the corresponding publicly verifiable signature, the message is no longer linkable to the receiver. Thus the receiver or any third party can use it for malicious purposes. On the other hand, this is not the case in our SS scheme as the (special) signature generated by a signer is given to a specific receiver.

Undeniable signature [8] and designated confirmer signature [5]: In the former scheme, the recipient has to interact with the signer to be convinced of its validity whilst in the latter scheme, the signatures has to be verified by interacting with an entity, the confirmer, designated by the signer. Both signature schemes require an interactive protocol to carry out signature verification. In our proposed SS scheme only a simple and computationally cheap algorithm is required to carry out the signature verification.

Nominative signature [16] scheme: This scheme allows a nominator (signer) and a nominee (verifier) to jointly generate and publish the signature in such a way that only the nominee can verify the signature and if necessary, only the nominee can prove to a third party that the signature is valid. Although signature privacy can be achieved using this scheme, it requires an interactive protocol for the signing stage. In our proposed SS scheme, a signer is able to generate the signature on his/her own.

Designated Verifier Signature (DVS) scheme: This scheme, independently proposed by Jakobsson, Sako, and Impagliazzo [15][4] and Chaum [6] in 1996, provides signature privacy. Although the designated verifier can be convinced of the validity of the signature in the DVS scheme, the verifier is unable to transfer the conviction to any other entity. A major difference between our proposed SS scheme and the DVS scheme is that the latter is unable to provide public provability of the signature.

[4] Lipmaa, Wang, and Bao [19] pointed out a weakness in this DVS scheme [15].

Limited Verifier Signature (LVS) scheme: This scheme, first proposed by Araki, Uehara, and Imamura in 1999 [1], differs from the DVS scheme in that the limited verifier is able to transfer the proof to convince another entity (e.g., a judge) if the signer has violated some rule non-cryptographically. Such a proof is, however, not transferrable to a third entity. In 2004, Chen *et. al.* proposed a convertible limited verifier signature scheme in a pairing-based setting [9]. Their scheme allows the limited verifier signature to be transformed into a publicly verifiable signature. This converted limited verifier signature, however, is rather loosely related to the limited verifier any more since the limited verifier is unable to prove that he is the intended recipient of the converted signature. On the other hand in our proposed SS scheme, any receiver can prove publicly that he is the legitimate receiver of the corresponding signature.

Anonymous signature scheme: First proposed by Yang *et al.* [25], this scheme appears to have similar property. However, the anonymous signature scheme provides signer anonymity and does not have an intended receiver when the signature is generated. We will provide an example in Section 7 to better explain this difference.

Signcryption scheme [26]: This scheme, first proposed by Zheng in 1997, is perhaps most similar to our proposed SS scheme. The signcryption scheme is built from a clever combination of a secure encryption scheme and a secure signature scheme, providing both confidentiality and non-repudiation. Many extensions of the signcryption scheme have also been proposed (e.g., [2, 3, 18, 22, 23][5]). Signcryption provides signature privacy by encrypting the message. However, this is computationally expensive particularly for applications that do not require message confidentiality. SS is a new approach to provide signature privacy without encryption.

At first read, our proposed SS scheme might be confused with other previously published signature privacy-related signature schemes. However, if we refer to the definition of SS scheme given in Definition 1, it is clear that:

- The undeniable signature scheme differs from the SS scheme since interactive protocol between the signer and the verifier is required in the signature verification algorithm.
- DVS and LVS schemes differ from the SS scheme since public provability cannot be achieved.
- Convertible LVS scheme differs from the SS scheme since the converted signature is not related with the receiver in any way.
- The anonymous signature scheme differs from the SS scheme since the signature does not have an intended receiver when the signature is generated whilst in our proposed SS scheme, an intended receiver is required at the time of signature generation.

On the other hand, the signcryption scheme is most similar to our proposed SS scheme if the public proving/verification algorithms are further defined.

[5] The signcryption scheme of Libert and Quisquater [18] is shown to be insecure [24].

6 Comparison of Efficiency

Since the signature-privacy-related signature schemes presented in Section 5 – with the exception of the signcryption scheme – have rather different functionalities, we will restrict our comparison only to the signcryption scheme. Since public proving/verification protocols were not defined in the original signcryption scheme, we assume that similar zero-knowledge proof techniques are applied in order to facilitate our comparison. Note that both the general and anonymous proofs are also possible in the signcryption scheme.

For completeness, we now describe briefly how the general and anonymous proofs are possible in Zheng's signcryption scheme. To prove the correctness of signcryption, either the signer or the receiver has to compute and reveal the following information (we use the same notation as the original paper).

- The signer, A, has to keep the random number, x, used in the signcryption secret. A has to compute and reveal y_b^x and g^x (requires 1 extra E). In this case, the public verifier has to check whether $g^x \stackrel{?}{=} g^{rs} y_a^s$ holds (requires $2E$).
- The receiver, B, has to compute and reveal $g^x = g^{rs} y_a^s$ and $y_b^x = (g^x)^{x_b}$ (requires $3E$).

Now, using the general proof protocol, the signer can then prove $ZKP(x) : (\log_g g^x = \log_{y_b} y_b^x = x)$ and the receiver can prove $ZKP(x_b) : (\log_g y_b = \log_{g^x} y_b^x = x_b)$, which requires $2E$ for proof and $4E$ for verification. Anonymous proof is also possible for the $\langle g, g^x, y_b, y_b^x \rangle$ tuple, which requires $6E$ for proof and $8E$ for verification.

Some of the computations required by the signer in our scheme can be performed offline (i.e., before the message to be sent and the receiver are known), such as $U = g^{r_A}$, and hence, provides efficiency. In Table 1, we compare the efficiency of our scheme with Zheng's signcryption scheme. The notation used in Table 1 is as follows: E and E_{off} denotes online and offline modular exponentiations, respectively; and ED denotes the cost for symmetric encryption or decryption. For simplicity, we ignore modulo multiplication/division operations and hash operations in our comparison.

Compared with Zheng's signcryption, our proposed SS scheme is more efficient both in the actual signature scheme and the public proving. Since the SS scheme does not use symmetric encryption, it is more efficient than signcryption, especially with long message when message confidentiality is not required. In public proving, the same zero-knowledge proofs can be used. In signcryption, however, both the signer and receiver have to compute and reveal additional information. All required information is already included in the generated signature in the SS scheme. Therefore, the SS scheme is more efficient than signcryption when used in business transactions that do not require confidentiality of messages.

		Signcryption [26]	Proposed SS scheme
Signing		$1E + 1ED$	$1E + 1E_{off}$
Verification		$2E + 1ED$	$3E$
General proof by signer	Proof	$3E$	$2E$
	Verification	$6E$	$4E$
General proof by receiver	Proof	$5E$	$2E$
	Verification	$4E$	$4E$
Anonymous proof by signer	Proof	$7E$	$6E$
	Verification	$10E$	$8E$
Anonymous proof by receiver	Proof	$9E$	$6E$
	Verification	$8E$	$8E$
Public verification		$1ED$	$2E$

Table 1. Summary of efficiency analysis

7 Applications of Secret Signatures

The proposed SS scheme can be used as an important cryptographic primitive to achieve business privacy, providing both signature privacy and public provability of signature efficiently.

Secret signature is useful in many applications where (1) the privacy of generated signature needs to be maintained, but the authorship of the signature can be publicly proven at a later stage, (2) message confidentiality is not very important, and (3) efficiency is critical.

We now describe some possible application examples.

Application 1: Private Business Transaction

Let's assume that two business entities, A and B, wanting to exchange some not-so-confidential contract document, m (e.g., m can be constructed using open source information). Although the contents of m do not need to be confidential, both A and B do not want to reveal to other entities that they had signed m. By using the proposed SS scheme, both A and B are assured that no third party is aware that m has been signed. In the event that one of the entities violates a non-cryptographic business rule, the other entity can prove the validity of their private contract to a third party by opening the generated secret signature. The public provability property guarantees the fairness of private business.

Application 2: Public Auction

We consider an application is a public auction (or English auction) setting where bidding prices are published and the bidders are allowed to bid prices anonymously as frequently as desired. When the winner is finally decided, the anonymity of the winning bid is revealed and the correctness of the winning bid

should be proven publicly (to provide public verifiability). This is a typical example where message confidentiality is not required, but signature privacy and public provability are required.

To provide the anonymity of bid with public provability, we can use the proposed SS scheme. In the bidding stage, bidders bid their prices using a secret signature with the auctioneer as a receiver. For example, let A be the auctioneer, B_i be bidders, and p_j be the bidding prices. Bidder B_i computes

$$s_{i,j} = SS.Sign(\mathsf{params}, p_j, sk_{B_i}, pk_A), \quad k_{i,j} = E_{pk_A}(B_i),$$

and posts $\langle p_j, s_{i,j}, k_{i,j} \rangle$ on the bulletin board, where $k_{i,j}$ is an encrypted ID information of the bidder (can be decrypted only by the auctioneer). In the winner announcement stage, the auctioneer opens the highest price bid and proves the correctness of secret signature. Any misbehavior of the auctioneer or bidders can be proven publicly. Note that bidder anonymity was achieved easily without using any encryption. Also note that threshold cryptography can be used by the auctioneer such that pk_A is distributed to multiple auctioneers and bidder information of the losing bids is kept secret.

Application 3: Paper Submission System

Consider a paper submission system for a conference. In this case submitted papers should be anonymous while they need not be encrypted. Authors need to commit the following facts to the program chair with a signature; (1) the submitted paper is their authentic work, (2) the submitted paper is not submitted in parallel to another conference or journal, (3) the authors will present the paper if accepted, and etc. Upon receiving the submission, the program chair has to issue a receipt for the submitted paper to the author.

In such an application, the authors and the conference program chair can exchange secret signatures with the other entity as a receiver (or publish the secret signature on the bulletin board as a public commitment). If general signatures are exchanged, anonymity will be broken (since anyone can verify the authorship of the submitted paper using the respective public keys). In the unlikely event of a dispute between the program chair and an author at a later stage, it can be resolved easily by using the public proving feature of secret signature.

One may note that the generated secret signature is tightly bound to the recipient, the program chair of the conference. Hence, the same signature cannot be submitted to another recipient, the program chair of another conference. Therefore, if it was subsequently discovered that the same paper with two different signatures was submitted to two different conferences in parallel, the author cannot repudiate his misbehavior. However, this is not the case for Yang *et al.*'s [25] anonymous signature scheme; the generated signature is not bound to any recipient. The signer is able to repudiate that someone other than the signer has forwarded the signature and the paper to the program chair of another conference without the signer's knowledge.

8 Conclusion

We had discussed the signature privacy issue and introduced a new signature variant, which we termed secret signature (SS). The SS scheme provides signature privacy and public provability of signature in an efficient manner by combining secure signature schemes and non-interactive one-way key agreement schemes. Although this is a very simple concept, it is a useful and efficient cryptographic tool to achieve business privacy.

We then presented a concrete implementation example of secret signature in discrete logarithm-based cryptosystems. Future extension of this work includes implementing the SS scheme in other cryptosystems such as RSA-based and pairing-based cryptosystems.

9 Acknowledgement

The second author would like to thank Sherman SM Chow for pointing out references [10] and [17].

References

1. S. Araki, S. Uehara, and K. Imamura. The Limited Verifier Signature and its Applications. *IEICE Transactions*, E82-A(1):63–68, 1999.
2. J. Baek, R. Steinfeld, and Y. Zheng. One-time Verifier-based Encrypted Key Exchange. In *PKC 2002*, volume 2274/2002 of *LNCS*, pages 80–98. Springer-Verlag, 2002.
3. F. Bao and R. H. Deng. A Signcryption Scheme with Signature Directly Verifiable by Public Key. In *PKC 1998*, volume 1431/1998 of *LNCS*, pages 55–59. Springer-Verlag, 1998.
4. M. Bellare and P. Rogaway. Random Oracles Are Practical: A Paradigm For Designing Efficient Protocols. In *ACM CCS 1993*, pages 62–73. ACM Press, 1993.
5. D. Chaum. Undeniable Signatures. In *EUROCRYPT 1994*, volume 950/1995 of *LNCS*, pages 86–91. Springer-Verlag, 1994.
6. D. Chaum. Private Signature and Proof Systems. United States Patent 5,493,614, 1996.
7. D. Chaum and T. P. Pedersen. Wallet Databases with Observers. In *CRYPTO 1992*, volume 740/1993 of *LNCS*, pages 89–105. Springer-Verlag, 1992.
8. D. Chaum and H. van Antwerpen. Undeniable Signatures. In *CRYPTO 1989*, volume 435/1990 of *LNCS*, pages 212–216. Springer-Verlag, 1990.
9. X. Chen, F. Zhang, and K. Kim. Limited Verifier Signature Scheme from Bilinear Pairings. In *ACNS 2004*, volume 3089/2004 of *LNCS*, pages 135–148. Springer-Verlag, 2004.
10. S.S.M. Chow, S.M. Yiu, L.C.K. Hui, and K.P. Chow. Efficient Forward and Provably Secure ID-Based Signcryption Scheme with Public Verifiability and Public Ciphertext Authenticity. In *ICISC 2003*, volume 2971/2003 of *LNCS*, pages 352–369. Springer-Verlag, 2003.
11. W. Diffie and M. Hellman. Multiuser Cryptographic Techniques. In *AFIPS 1976 National Computer Conference*, pages 109–112. AFIPS Press, 1976.

12. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO 1986*, volume 263/1987 of *LNCS*, pages 186–194. Springer-Verlag, 1986.
13. S. D. Galbraith and W. Mao. Invisibility and Anonymity of Undeniable and Confirmer Signatures. In *CT-RSA 2003*, volume 2612/2003 of *LNCS*, pages 80–97. Springer-Verlag, 2003.
14. S. Goldwasser, S. Micali, and R. L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
15. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated Verifier Proofs and Their Applications. In *EUROCRYPT 1996*, volume 1070/1996 of *LNCS*, pages 321–331. Springer-Verlag, 1996.
16. S.J. Kim, S.J. Park, and D.H. Won. Zero-Knowledge Nominative Signatures. pages 380–392, 1996.
17. C.K. Li, G. Yang, D.S. Wong, X.Deng, and S.S.M. Chow. An Efficient Signcryption Scheme with Key Privacy. In *EuroPKI 2007*, volume 4582/2007 of *LNCS*, pages 78–93. Springer-Verlag, 2007.
18. B. Libert and J.-J. Quisquater. Efficient Signcryption with Key Privacy from Gap Diffie-Hellman Groups. In *PKC 2004*, volume 2947/2004 of *LNCS*, pages 187–200. Springer-Verlag, 2004.
19. H. Lipmaa, G. Wang, and Feng Bao. Designated Verifier Signature Schemes- Attacks, New Security Notions and a New Construction. In *ICALP 2005*, volume 1853/2000 of *LNCS*, pages 459–471. Springer-Verlag, 2005.
20. W. Mao. *Modern Cryptography: Theory and Practice*. Prentice Hall PTR, 25 July, 2003.
21. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13:361–396, 2000.
22. J.-B. Shin, K. Lee, and K. Shim. New DSA-Verifiable Signcryption Schemes. In *ICICS 2002*, pages 35–47. Springer-Verlag, 2002.
23. R. Steinfeld and Y. Zheng. A Signcryption Scheme Based on Integer Factorization. In *ISW 2000*, volume 1975/2002 of *LNCS*, pages 308–322. Springer-Verlag, 2000.
24. G. Yang, D. S. Wong, and X. Deng. Analysis and Improvement of a Signcryption Scheme with Key Privacy. In *ISC 2005*, volume 3650/2005 of *LNCS*, pages 218–232. Springer-Verlag, 2005.
25. G. Yang, D. S. Wong, X. Deng, and H. Wang. Anonymous Signature Schemes. In *PKC 2006*, volume 3958/2006 of *LNCS*. Springer-Verlag, 2006.
26. Y. Zheng. Digital Signcryption or How to Achieve Cost (Signature & Encryption) << Cost (Signature) + Cost (Encryption). In *CRYPTO 1997*, volume 1294/1997 of *LNCS*, pages 165–1793. Springer-Verlag, 1997.

A Proving the Equality of Two Discrete Logarithms

Let α and β be two independent generators of order q in modular p. A prover P tries to prove to a verifier V that two numbers $a = \alpha^x$ and $b = \beta^x$ have the same exponent without exposing x. We denote this proof as

$$ZKP(x) : (\log_\alpha a = \log_\beta b = x).$$

Based on the scheme by Chaum and Pedersen [7] and Fiat-Shamir's heuristics [12] the non-interactive proof can be done as follows.

$\underline{ZKP(x) : (\log_\alpha a = \log_\beta b = x)}$

- Proof: Prover P randomly chooses t from $\mathbb{Z}_q^*$ and computes $c = \alpha^t$ and $d = \beta^t$. He computes $h = \mathcal{H}(\alpha, \beta, a, b, c, d)$ and $s = t + hx$, then sends (c, d, s) to the verifier. Proof requires two exponentiation operations.
- Verification: Verifier V first computes $h = \mathcal{H}(\alpha, \beta, a, b, c, d)$. Then he checks $\alpha^s \stackrel{?}{=} ca^h$ and $\beta^s \stackrel{?}{=} db^h$. Verification requires four exponentiation operations.

B OR Proving the Equality of Two Discrete Logarithms

Let α_1, β_1, α_2, β_2 be four independent generators of order q in modular p.

A prover P tries to prove to a verifier V that either $\log_{\alpha_1} a_1 = \log_{\beta_1} b_1 (= x_1)$ or $\log_{\alpha_2} a_2 = \log_{\beta_2} b_2 (= x_2)$ holds using his knowledge of x_1 or x_2 without exposing it. It is an OR combination of two proofs for the equality of two discrete logarithms. We denote this proof as

$$ZKP(x_1 \vee x_2) : (\log_{\alpha_1} a_1 = \log_{\beta_1} b_1 = x_1) \vee (\log_{\alpha_2} a_2 = \log_{\beta_2} b_2 = x_2).$$

The prover knows either x_1 or x_2, but does not know them all. This proof can be done as follows.

$\underline{ZKP(x_1 \vee x_2) : (\log_{\alpha_1} a_1 = \log_{\beta_1} b_1 = x_1) \vee (\log_{\alpha_2} a_2 = \log_{\beta_2} b_2 = x_2)}$

- Proof: Assume that the prover P knows x_b and does not know $x_{b'}$.
  - Randomly chooses $r_b, s_{b'}, t_{b'}$ from $\mathbb{Z}_q^*$.
  - Computes $c_b = \alpha_b^{r_b}, d_b = \beta_b^{r_b}, c_{b'} = \alpha_{b'}^{s_{b'}} a_{b'}^{t_{b'}}, d_{b'} = \beta_{b'}^{s_{b'}} b_{b'}^{t_{b'}}$ (incurring six exponentiation operations).
  - Computes $t = \mathcal{H}(\alpha_1, \beta_1, \alpha_2, \beta_2, a_1, b_1, a_2, b_2, c_1, d_1, c_2, d_2)$.
  - Computes $t_b = t - t_{b'}$ and $s_b = r_b - t_b x_b$, then sends $(c_1, d_1, c_2, d_2, s_1, t_1, s_2, t_2)$.
- Verification: Verifier V checks
  - $c_1 \stackrel{?}{=} \alpha_1^{s_1} a_1^{t_1}$, $d_1 \stackrel{?}{=} \beta_1^{s_1} b_1^{t_1}$, $c_2 \stackrel{?}{=} \alpha_2^{s_2} a_2^{t_2}$, $d_2 \stackrel{?}{=} \beta_2^{s_2} b_2^{t_2}$ and
  - $t_1 + t_2 \stackrel{?}{=} \mathcal{H}(\alpha_1, \beta_1, \alpha_2, \beta_2, a_1, b_1, a_2, b_2, c_1, d_1, c_2, d_2)$.