

Constructing parallel long-message signcryption scheme from trapdoor permutation

HU ZhenYu^{1,2†}, LIN DongDai¹, WU WenLing¹ & FENG DengGuo¹

¹ State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China;

² Graduate School of the Chinese Academy of Sciences, Beijing 100039, China

A highly practical parallel signcryption scheme named PLSC from trapdoor permutations (TDPs for short) was built to perform long messages directly. The new scheme follows the idea “scramble all, and encrypt small”, using some *scrambling operation* on message m along with the user’s identities, and then passing, in parallel, small parts of the scrambling result through corresponding TDPs. This design enables the scheme to flexibly perform long messages of arbitrary length while avoid repeatedly invoking TDP operations such as the CBC mode, or verbosely black-box composing symmetric encryption and signcryption, resulting in noticeable practical savings in both message bandwidth and efficiency. Concretely, the signcryption scheme requires exactly one computation of the “receiver’s TDP” (for “encryption”) and one inverse computation of the “sender’s TDP” (for “authentication”), which is of great practical significance in directly performing long messages, since the major bottleneck for many public encryption schemes is the excessive computational overhead of performing TDP operations. Cutting out the verbosely repeated padding, the newly proposed scheme is more efficient than a black-box hybrid scheme. Most importantly, the proposed scheme has been proven to be tightly semantically secure under adaptive chosen ciphertext attacks (IND-CCA2) and to provide integrity of ciphertext (INT-CTXT) as well as non-repudiation in the random oracle model. All of these security guarantees are provided in the full multi-user, insider-security setting. Moreover, though the scheme is designed to perform long messages, it may also be appropriate for settings where it is impractical to perform large block of messages (i.e. extremely low memory environments such as smart cards).

authenticated encryption, signcryption, trapdoor permutations, parallel

Received December 28, 2005; accepted May 22, 2006

doi: 10.1007/s11432-007-2018-x

†Corresponding author (email: zhenyu@iscas.cn)

Supported by the National Basic Research Program (Grant No. 2004CB318004), and the National Natural Science Foundation of China (Grant Nos. 60373047 and 90604036)

1 Introduction

1.1 Background

Traditionally, cryptographic operations for signature and encryption have been considered as separate building blocks (“primitives”) that serve in distinct security purposes—the former is to guarantee the authenticity of messages and the latter is to the privacy of messages. In practice, however, many important applications, for example, secure electronic mail, require both authenticity and privacy. Applications of such kind would generally compose signature and encryption directly to achieve these goals. However, simply jointing an encryption block and an authentication block does not necessarily achieve both of the two security requirements^[1]. Thus, it seems justified to invest special effort into designing a tailored and efficient solution to implement a “joint signature and encryption” *primitive*. This observation motivated the study of a new cryptographic primitive known as signcryption, which is designed to achieve simultaneous privacy and authenticity in the public key cryptographic setting.

Signcryption was first introduced by Zheng^[2] in 1997, with the expressed goal of achieving a signcryption scheme with a smaller “cost” than a sequential composition of signature and encryption. Thereafter, several papers^[3–5] offered security arguments about various signcryption schemes, and only recently Baek et al.^[6] and An et al.^[1] made on this subject the first formal investigations. Both of them define signcryption as a multi-user primitive, which simultaneously satisfies chosen ciphertext security for privacy and existential unforgeability for authenticity. Baek et al.^[6] showed that the original “discrete log-based” proposal of Zheng^[2] indeed can be proven secure in the random oracle (RO) model under the so-called Gap Diffie-Hellman assumption. An et al.^[1] examined generic composition methods of building signcryption from any secure signature and encryption, giving rise to a variety of signcryption schemes so that users can easily change their public keys or their favorite signature/encryption scheme, and still be able to seamlessly communicate with other users. While Zheng’s signcryption scheme is quite elegant and achieves very good efficiency in the discrete-log setting, it has the disadvantage that all parties must agree on the same public parameters (e.g. the common discrete log group). Thus, for example, any change to the security parameter or signcryption scheme requires consensus. On the other hand, while the generic schemes presented in ref. [1] validate that signcryption can be built from ordinary signature and encryption, their efficiency and security are still suspectable.

In practice, many truly-efficient signature and encryption schemes, such as OAEP^[7], OAEP+^[8], PSS-R^[9], are typically built from trapdoor permutations (TDPs), such as RSA, and are analyzed in the RO model. Even with these efficient implementations, when jointing, however, they may not certainly form an efficient or secure signcryption scheme^[1]. For example, users have to maintain independent keys for signature and for encryption, the message bandwidth is suboptimal (due to two independent “paddings” and additional overhead for identity fraud protection), and the scheme’s “exact security” is not tight. Moreover, many of the TDP-based public-key encryption schemes conduct the length of messages not to exceed the length of the key domain. If one wants to encrypt bulk data, a file for instance, he has to either iteratively invoke the primitive (e.g. in CBC or CTR mode, which is terrible in time cost because of repeated TDP operations), or use the KEM-DEM hybrid technique to combine key encapsulation mechanism (KEM) and data encapsulation mechanism (DEM)^[10,11]. Both of them regard the primitive (either asymmetric or symmetric) as “black-box”, and inevitably repeatedly pad (in both of the encryption primitives). Thus,

considering that practical schemes are already built from trapdoor permutations, and always use padding techniques, this paper, inspired by the idea of “scramble all, and encrypt small”, presents an optimized signcryption construction built from trapdoor permutations (in the RO model) that resolves the inefficiencies of the “black-box” composition, which shares features such as simplicity, efficiency, near-optimal exact security, and, most importantly, flexible length of messages.

1.2 Related works

While many of the popular signcryption schemes are based on TDP, we would like to compare our scheme to the most relevant previous works of refs. [1, 12, 13].

We think over the TDP-based implementation of the “commit-then-encrypt-and-sign” (CtE&S for short) from ref. [1]. In CtE&S, one first modifies the original message m into m' by concatenating it with the user’s IDs (identities); then applies a commitment scheme to transform the modified message m' into a pair $\langle d, c \rangle$, and then encrypts d and signs c . Notice that, for the encryption and signature, one may usually apply two new, independent padding schemes (such as RSA-OAEP^[7,8], PSS-R^[9]) to d and c to obtain w and s , only then applies corresponding TDPs to w and s , and get the final ciphertext. Thus, the message is padded 4 times (hash of keys, commitment, signature and encryption), which may be possibly hashed 7 times (one for hashing keys, two for commitment, and two for signature and encryption respectively). In fact, for currently best-known TDP-based encryption methods, one either has to lose exact security^[8] or has to pad the message to be longer than the length of the TDP domain^[14] (CtE&S cannot achieve both INT-CTXT- and IND-CCA-security, but achieves slightly weaker notions; see ref. [1] for details). Even so, both lengths of the two components of commitment are confined not to exceed the corresponded key size, which in turn confines the message length not to exceed the sum of the two key sizes. In contrast, we just pad once (if we regard it as padding analogically) which hashes only twice, and more importantly, we are always guaranteed to obtain INT-CTXT and IND-CCA2 security for the insider security, and can perform the messages of arbitrary length.

Dodis et al.^[12] had proposed a generic parallel padding scheme called PSEP (Probabilistic Signature and Encryption Padding) for signcryption scheme based on secure commitment and the famous Feistel Transform^[15]. The main idea is that, to signcrypt a message m , one first commits the original message into two sub-messages $\langle d, c \rangle$ by the specific commit scheme:

$$\begin{aligned} d &= m_2 || r, \\ c &= (m_1 \oplus \mathcal{H}(r)) || \mathcal{H}(m_2 || r). \end{aligned}$$

Then apply a single round of Feistel Transform to $\langle d, c \rangle$ to get the padding output $\langle w, s \rangle$. The final signcryption ciphertext is got by applying two trapdoor permutations, which stand for signing and encrypting operation, to w and s respectively.

Although the author claimed that, for the public key domain length of k , the PSEP scheme potentially allows one to fit $|m| \approx 2k$, it is not practical, since, to make the $2^{-(k-|m_i|)}$ being negligible against a guessing attack, it is needed that $k - |m_i| \geq 160$, which means $|m| = |m_1| + |m_2| \leq 2(k - 160)$. In our scheme, we do not restrict the message to a certain length. Instead, the efficiency of our scheme goes up with the increasing of message length not only in the bandwidth but also in the time expenditure. If we keep the public key length being 1024, then the message length 1024 enables the bandwidth of our scheme to be 33%; but when the message length is doubled, the bandwidth is up to 50%, and when the message length is up to 8 times of the key length, the bandwidth can be up to 80%. The longer the message is, the higher the bandwidth, while the total

computing expenditure keeps almost unvaried, considering that the hashing operations are much faster and the total time consumption is dominated by the TDP operations.

To deal with long messages, Dodis^[12] also suggested using the black-box hybrid method to extend any short-message signcryption scheme with support for associated data to include support for long messages. Given the OTP (one time padding) encryption and PSEP suggested by the authors, the long-message signcryption scheme of ref. [12] takes 4 hashes (one for OTP operation, two for commitment and one for Feistel transformation). Comparing with ours, it spends 2 more hashes, without extra gains in bandwidth.

We compare our scheme with RSA-TBOS^[13]. This is a signcryption scheme that uses PSS-R padding for sequential composition of RSA signature and encryption. Namely, to transmit $RSA_R(RSA_S^{-1}(w||s))$, where $w||s$ is the result of PSS-R applied to the message m , and RSA_U and RSA_U^{-1} are the RSA and inverse RSA function respectively under the key of user U . Unfortunately, PSS-R does not happen to be a good padding scheme as expected. For example, their results do not imply practical security guarantees even when using a 2048-bit RSA modulus. In addition, the RSA-PSS-R padding scheme for encryption has a rather low bandwidth or message recovery: the size of the recovered message must be below half the size of the modulus. In the typical key setting of $k = |N| = 2048$ and $k_0 = 160$, we can obtain as maximum $|m| = 826$, that is, $|m|$ is only up to 42% of $|N|$. Comparatively, our scheme provides tightly exact security, supports flexible length of messages, and is more efficient (see section 4 for details). Furthermore, it can be processed in parallel, which can get more gains than the RSA-TBOS.

1.3 Our contributions

Our scheme is built from the usually TDPs. In our scheme each user U independently picks a *single* trapdoor permutation f_U (together with its trapdoor, denoted by f_U^{-1}) and publishes f_U as its public signcryption key. Unlike generic TDP-based signature and encryption schemes, our scheme follows the idea “scramble all, and encrypt small”, using some *scrambling operation* on message m along with the user’s IDs, and then passing small parts of the scrambling result through corresponding TDPs. Our scheme uses totally only one *single* specialized scrambling operation, rather than one independent scrambling for each TDP computation. This design results in noticeable practical savings in both quantitative and qualitative security, as well as improves the message bandwidth and efficiency. Note that we do not claim any improvement in the *computational* efficiency of TDPs, instead, we manage to reduce the invocation of TDPs for long message, since in practice the computational overhead is completely dominated by the time required to compute and invert TDPs. In particular, our signcryption schemes will require exactly one computation of the “receiver’s TDP” (for “encryption”) and one inverse computation of the “sender’s TDP” (for “authentication”), which is clearly optimal for TDP based signcryption in dealing with long message. Moreover, avoiding verbosely combining symmetric and asymmetric primitives, the performance of our scheme is even better than the black-box hybrid scheme that is widely used for long message signcryption.

The scheme we proposed in this paper is based on the work of Dodis et al.^[12,16], who present a paradigm known as padding-based parallel signcryption. While the primary computational advantage of padding-based parallel signcryption over straightforward composition of encryption and signature lies in its parallelization of the signature and encryption operations (which is rarely of practical value), the advantages of the scheme in terms of security, flexibility, and compatibility are

sufficient to merit its use. For a detailed comparison of the advantage of padding-based parallel signcryption in relation to other signcryption schemes, see ref. [16]. Most importantly, our proposed scheme has been proven to be semantically secure under adaptive chosen ciphertext attacks (IND-CCA2) and to provide integrity of ciphertext (INT-CTXT). Both of these security guarantees are provided in the full multi-user, insider-security setting (described in further detail below). Our proposed signcryption scheme also includes support for long messages (such as emails).

Finally, we evaluate the performance of our scheme in the context of different length of messages. Our implementation is based on the Crypto++4.2 library, modified to include support for twice RSA operations and once scrambling operation. Also, for comparative purposes, we use the same RSA operations and hash schemes as RSA-OAEP implemented in Crypto++4.2, and keep keys the same as RSA-OAEP does. We show that substantial performance improvements can be realized by encrypting messages of large bulk.

1.4 Overview of this paper

Section 2 presents some preliminaries used in this paper; section 3 provides the construction of the new scheme and its security; sections 4 and 5 discuss its performance and some other features. The conclusion is produced in section 6.

2 Preliminaries

2.1 Notations

Throughout this paper, we will use the symbol “ $|x|$ ” to denote the bit length of x , “ $x||y$ ” to the concatenation of x and y . The symbol “ \oplus ” denotes the bitwise-exclusive-OR operation. If n is a positive integer, then the symbol $\{0,1\}^n$ denotes the set of n -bit binary strings (We also use the symbol $\{0,1\}^*$ to denote the set of binary strings with no fixed length). If f is a randomized (resp., deterministic) algorithm, then “ $x \xleftarrow{R} f(y)$ ” (resp., “ $x \leftarrow f(y)$ ”) denotes the process of running f on input y and assigning the result to x . However, if M is a set, then “ $x \xleftarrow{R} M$ ” denotes that x is randomly chosen from M .

2.2 Trapdoor permutations

A family of trapdoor permutations (TDPs) is a family of one-way permutations with an extra property. There exists a secret inverse function (the trapdoor) for each member f of the family that allows its possessor to efficiently invert f at any point in the domain of his choosing. It should be easy to randomly generate a permutation f and some “trapdoor” associated with f . Furthermore, f is easy to compute and, given the trapdoor information, so is its inverse f^{-1} . However, without the trapdoor, f is “hard” to invert on any inputs, that is, for any adversary \mathcal{A} , given $y = f(x)$ for random x , the probability of successfully finding x is negligible in the security parameter k of the generation algorithm. A TDP family F is said ϵ_{TDP} -secure, if for any member f_i of F and any image $f_i(x)$, no PPT (Probabilistic polynomial time) adversary \mathcal{A} can, without the correct trapdoor, successfully find the pre-image x with probability greater than ϵ_{TDP} .

2.3 Signcryption

In this section, we formally describe multi-party signcryption and its security. We generalize the definitions of refs. [1, 17] to include support for user’s identities, in order to provide more useful

functionality and more general results.

2.3.1 Syntax of signcryption scheme. A signcryption scheme consists of the algorithms $\mathcal{SC} = (\mathcal{K}, \mathcal{SE}, \mathcal{VD})$. In the multi-party setting, the $\mathcal{K}(k)$ algorithm for user U generates the key-pair (pk_U, sk_U) , where k is the security parameter, sk_U is the signing/decryption key that is kept private, and pk_U is the verification/encryption key made public. Without loss of generality, we assume that pk_U is determined by sk_U .

The randomized signcryption (sign/encrypt) algorithm \mathcal{SE} takes as input the sender S 's public key pk_S , secret key sk_S , the receiver R 's public key pk_R , and a message m from the associated message space \mathbf{M} , and internally some coins and outputs a signcryption (ciphertext) c ; we write $c \leftarrow \mathcal{SE}_{pk_S, sk_S, pk_R}(m)$ (sometimes, we omit pk_S, sk_S, pk_R and write $c \leftarrow \mathcal{SE}(m)$).

The deterministic de-signcryption (verify/decrypt) algorithm \mathcal{VD} takes as input the signcryption (ciphertext) c , the receiver R 's public key pk_R , secret key sk_R , the sender S 's public key pk_S , and outputs $m \in \mathbf{M} \cup \{\perp\}$, where “ \perp ” indicates that the message was not encrypted or signed properly. We write $m \leftarrow \mathcal{VD}_{pk_S, sk_R, pk_R}(c)$ (again, we can omit the keys and write $m \leftarrow \mathcal{VD}(c)$). We require that $\mathcal{VD}(\mathcal{SE}(m)) = m$, for any $m \in \mathbf{M}$.

Compared with the definitions of refs. [1, 17], the notion we present here is slightly different from theirs. Here, we include clearly the sender's and receiver's identifications (public keys) in both signcryption and de-signcryption for two reasons: First, both of the sender and receiver's identifications (public keys) are available to everyone, and can be used when encrypting and decrypting. Second, involving both parties' public keys causes the ciphertext to be bound to the genuine users, which may be used to improve the security of a signcryption scheme. For instance, considering the Encrypt-then-Sign scheme in the multi-user setting, a man-in-the-middle attacker may intercept a ciphertext, and strip off the signature of the original sender, then re-sign it using his own key and send to the intended receiver to declare himself to be the origin of the message. Binding the sender's public key into the ciphertext may resist this type of attack. Also for the Encrypt-then-Sign scheme, the receiver may re-choose a new pair of keys to make the same ciphertext to de-signcrypt into different messages. Binding the receiver's public key into the ciphertext may cause a failure during de-signcrypting, if the receiver changes his public-secret key pair.

2.3.2 Security of signcryption scheme. In this paper, we only use the strongest possible notion of *Insider security* for *multi-user* signcryption^[1]. As expected, the security for signcryption consists of IND-CCA2 (Indistinguishability under adaptive chosen-ciphertext attacks) and INT-CTXT (Integrity of ciphertext) components when attacking some user U .

For the purposes of this paper, we will consider a signcryption scheme to be secure when it is provably both IND-CCA2 and INT-CTXT secure against insiders in the multi-user setting. Security against insiders implies that the authenticity of the scheme is protected even if the recipient of the ciphertexts is in fact malicious, and that the privacy of the scheme is protected even if the sender of the ciphertexts is malicious. Clearly, this notion is a stronger notion of security than the outsider security (which does not provide these guarantees). Multi-user security implies that no “identity fraud” can occur in a setting where multiple users may have a key (which is generally the setting of interest). In this setting, IND-CCA2 security implies that any PPT adversary has at most negligible success advantage in the following game: The adversary is first allowed to query a user R to signcrypt arbitrary messages to arbitrary recipients, or to de-signcrypt arbitrary ciphertexts of the

adversary's choosing. The adversary then selects two messages, M_0 and M_1 , asks the challenger to signcrypt one of them at random (with R as the recipient, using a sending secret key of the adversary's choice), and to return the corresponding "challenge" ciphertext to him. The adversary may then continue to query R for arbitrary signcryptions and de-signcryptions, subject to the restriction that he may not request the de-signcrypt of the challenge ciphertext. In the end, the adversary outputs a bit b' as his own decision to indicate which one of M_0 and M_1 is encrypted. We say that \mathcal{A} "succeeds" if $b=b'$. Let $Adv_{SC,\mathcal{A}}^{ind-cca2}(RL) = |2 \cdot \Pr[b=b'] - 1|$ denote his advantage over the random guessing. Here RL is a list of variables specifying the resources of interest for the adversary in question. Adversarial resources to which we pay attention are: t , the running time of the adversary; q , the number of queries asked by the adversary; and μ , the aggregate length of these queries.

Definition 1 (IND-CCA2 secure). A signcrypt scheme SC is indistinguishability under adaptive chosen-ciphertext attacks, or IND-CCA2 secure, if for any PPT adversary \mathcal{A} that plays the above game, the advantage $Adv_{SC,\mathcal{A}}^{ind-cca2}(\cdot)$ is negligible with respect to the secure parameter.

Note that to deal with insider security, in Definition 1 the adversary is assumed to have the private key of the sender of a signcrypt message. This means that confidentiality is preserved even if a sender's key is compromised (i.e. forward security). This requirement is necessary in some case. For example, assume an adversary \mathcal{A} happens to steal the key of S , even though now \mathcal{A} can fake messages "from S to R ", the Insider-security can keep the adversary \mathcal{A} from understanding the previous (or even future) recorded signcryptions sent from honest S to R , while the Outsider-security might not.

The INT-CTXT security definition implies that any polynomial time adversary has at most negligible success advantage in the following game: The adversary is allowed to query user S to signcrypt messages of his choice to recipients of his choice, or de-signcrypt arbitrary ciphertexts of his choice. Eventually, the adversary attempts to output a valid ciphertext that appears to be sent by S , but was not previously returned to the adversary by one of his queries to S . Let $Adv_{SC,\mathcal{A}}^{int-ctxt}(RL) = \Pr[\text{success}]$ denote his advantage.

Definition 2 (INT-CTXT secure). A signcrypt scheme SC is integrity of ciphertext, or INT-CTXT secure, if for any PPT adversary \mathcal{A} that plays the above game, the advantage $Adv_{SC,\mathcal{A}}^{int-ctxt}(\cdot)$ is negligible with respect to the secure parameter.

To deal with the insider security, in Definition 2 the adversary is assumed to have the private key of the receiver of a signcrypt message. This means that the integrity of ciphertext is preserved even if a receiver's key is compromised. This requirement is to keep the adversary \mathcal{A} from generating valid signcryptions (by re-keying, for instance) that were not actually sent by honest S .

A signcrypt scheme satisfying the above security requirements represents an extremely resilient cryptographic primitive that may be safely used in a wide variety of contexts. As for the analysis in ref. [1], a simple composition of signature and encryption will not satisfy these requirements. In particular, any composition of PKCS#1^[18] encryption and signature schemes will not satisfy these definitions of security (even using the "commit then encrypt and sign" paradigm proposed in ref. [1]). In this paper, we propose the scheme that scrambles the whole message along with the user's IDs, then computes two TDPs in parallel and provides all these security guarantees.

3 The new scheme and its security

We first introduce the generic construction of our new scheme, which uses a general trap door permutation function family. To facilitate the description, we name our scheme “Parallel Long-message Signcryption” (PLSC for short). In the definition, we assume that the algorithm \mathcal{K} generates a $(t, \varepsilon_{\text{TDP}})$ -secure TDP pair (f_U, f_U^{-1}) given to the honest user U . The RSA and Rabin functions are the two best-known TDPs. For security intuition, it is also instructive to think of the sender S and the recipient R as acting maliciously (since we are in the Insider-security model^[11]). The encryption uses two cryptographic hash functions and takes as input a plaintext message, a random string of the length of the TDP domain, and for resisting identity fraud, the IDs of the sender and intended receiver. Figure 1 depicts the process. Detailed instructions are defined as follows.

Definition 3 (Construction of PLSC). Let n and l be two positive integers. Let $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^n$ and $\mathcal{G} : \{0,1\}^* \rightarrow \{0,1\}^l$ be two cryptographic hash functions. Let $F : \{0,1\}^l \rightarrow \{0,1\}^l$ be a trap door permutation family, ID be a public determinative algorithm that takes as input a user’s public key and outputs the user’s identification (it may simply be a hash function or the identity transformation). Define the signcryption scheme $SC=(\mathcal{K}, \mathcal{SE}, \mathcal{VD})$ that uses the hash function \mathcal{H}, \mathcal{G} and trap door permutation family F as follows:

<p>Algorithm $\mathcal{K}(k)$</p> $f_R \xleftarrow{R} F$ $f_S \xleftarrow{R} F$ <p>Return $\langle f_S, f_S^{-1} \rangle, \langle f_R, f_R^{-1} \rangle$</p>	<p>Algorithm $\text{SE}_{f_S, f_S^{-1}, f_R} (m)$:</p> $id_S \leftarrow \text{ID}(f_S)$ $id_R \leftarrow \text{ID}(f_R)$ $r \xleftarrow{R} \{0,1\}^l$ $p \leftarrow \mathcal{H}(r, id_S id_R)$ $g \leftarrow \mathcal{G}(m r, id_S id_R)$ $\sigma \leftarrow f_S^{-1}(g)$ $c_1 \leftarrow f_R(r)$ $c_2 \leftarrow m \oplus p$ $c \leftarrow c_1 c_2 \sigma$ <p>Return c</p>	<p>Algorithm $\text{VD}_{f_S, f_R^{-1}, f_R} (c)$:</p> $id_S \leftarrow \text{ID}(f_S)$ $id_R \leftarrow \text{ID}(f_R)$ <p>Parse c as $c_1 c_2 \sigma$</p> $r \leftarrow f_R^{-1}(c_1)$ $p \leftarrow \mathcal{H}(r, id_S id_R)$ $m \leftarrow c_2 \oplus p$ $g \leftarrow \mathcal{G}(m r, id_S id_R)$ $g' \leftarrow f_S(\sigma)$ <p>If $g = g'$ then return m else return \perp</p>
--	--	--

The function $\mathcal{G}(\cdot)$ can be any fixed length output collision resistant hash function (CRHF for short) that can be modeled as a random oracle that is defined in PKCS#1 standard (for example, the SHA256 hash function), with an output length suitable to the inversion TDP operation. The function $\mathcal{H}(\cdot)$ can use the MGF (mask generating function) that extends a fixed length output CRHF to a variable length output CRHF. The details to construct a MGF using CRHF are provided in the PKCS#1 v2.1 document^[18].

Theorem 1 (Security of PLSC). Let $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^n$ and $\mathcal{G} : \{0,1\}^* \rightarrow \{0,1\}^l$ be two cryptographic hash functions, $F : \{0,1\}^l \rightarrow \{0,1\}^l$ be a trap door permutation family of the ε_{TDP} -secure. Let $SC=(\mathcal{K}, \mathcal{SE}, \mathcal{VD})$ be the signcryption scheme constructed via Definition 3. For any PPT adversary \mathcal{A} against IND-CCA2 security and adversary \mathcal{B} against INT-CTXT security, which queries the random oracle \mathcal{H} , \mathcal{G} , the signcryption oracle \mathcal{SE} , the de-signcryption oracle \mathcal{VD} at most q_H, q_G, q_S , and q_D times respectively, and takes time approximately t . We have

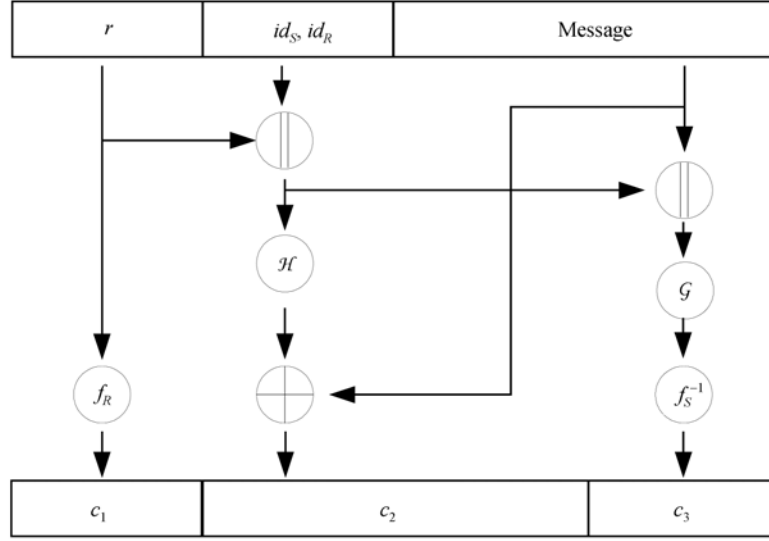


Figure 1 Construction of PLSC.

$$Adv_{SC,A}^{ind-cca2}(t, q_H, q_G, q_S, q_D) \leq q_D \cdot 2^{-n+1} + (q_D + q_S \cdot ((2^{-n} + 1)q_S + q_H + 2^{-n} \cdot q_G)) \cdot 2^{-l+1} + 4 \cdot \epsilon_{TDP}, \quad (1)$$

$$Adv_{SC,B}^{int-ctxt}(t, q_H, q_G, q_S, q_D) \leq (q_D + 1) 2^{-n} + (q_D + q_S \cdot ((2^{-n} + 1)q_S + q_H + 2^{-n} \cdot q_G) + 1) \cdot 2^{-l} + \epsilon_{TDP}, \quad (2)$$

$$t = O((q_G + 2 \cdot q_S + q_D)T_f), \quad (3)$$

where T_f is the time to compute a single TDP operation.

The proof of Theorem 1 is given in Appendix A.

4 Performance of the new scheme

To analyze the performance and compare with other schemes, we would like to take the RSA Trapdoor Permutations to create a signcryption scheme as in section 3. Under the strong RSA assumption, RSA is a family of strong one-way trapdoor permutations^[19]. So the RSA based signcryption scheme constructed via Definition 3 is still insider-secure in the sense of IND-CCA2 and INT-CTXT.

First, we evaluate the cost of computation of our scheme. Since the hash function can be computed very fast, the main cost of the scheme concentrates on the time expense of RSA (i. e. modular exponentiation). Let T_{exp} be the computational cost of a modular exponentiation, T_h a hashing, T_r a random string chosen from $\{0, 1\}^l$. Let $T_{se-PLSC}$ and $T_{vd-PLSC}$ be the total computational cost of the signcryption and de-signcryption respectively. We have

$$T_{se-PLSC} = T_r + 2T_h + 2T_{exp},$$

$$T_{vd-PLSC} = 2T_h + 2T_{exp}.$$

We compare it with the RSA-TBOS scheme^[13] that applies the PSS-R padding scheme. In the PSS-R-Padding, in order to guarantee that the padding result as an integer less than N_A , a trial-and-error test has to be conducted. Let $T_{se-rsa-tbos}$ and $T_{vd-rsa-tbos}$ be RSA-TBOS's total computational cost of the signcryption and de-signcryption respectively, if the leading-0 technique is used in PSS-R padding, then

$$T_{se-rsa-tbos} = T_r + 2T_h + 2T_{exp},$$

which is the same as our scheme.

Notice that, when signcryption, if $\sigma = \text{Padding}(m, r)^{d_A} \pmod{N_A} > N_B$, in order to produce the proper ciphertext, the most significant bit of σ must be chopped off, which can be fixed back during de-signcryption by the receiver's verification step. This occurs with roughly 1/2 probability. So to correctly recover the plaintext, it will be 1/2 probability to repeat the de-signcryption and the mean number of de-signcryption is 1.5. Thus

$$T_{vd-rsa-tbos} = T_{exp} + 1.5(2 \cdot T_h + T_{exp}) = 3T_h + 2.5T_{exp},$$

which is obviously more expensive than ours, particularly in the computation of T_{exp} because of that the overall overhead is absolutely dominated by the cost of T_{exp} .

Then we need to compare the communication cost of the RSA-based PLSC with RAS-TBOS. In the RSA-TBOS, the RSA-PSS-R padding scheme has a rather low bandwidth for message recovery: the size of the recovered message must be below half the size of the modulus. In the typical key setting of $k = |N_U| = 2048$ and $k_0 = 160$ (where N_U denotes the RSA module), the maximum $|m|$ we can obtain is only up to 42% of N_U . However, in our RSA-based PLSC scheme, considering that $n = 2 \cdot l$, $|s| = l$, $|p| = |m| = n$, $|c_2| = |p| = n$, and $|c_1| = |\sigma| = l$, so $|m| = 0.5|c|$. That is, the bandwidth is 50% of the RSA modulus. Moreover, if we promote the ratio of n to l , the bandwidth will be even higher, for instance, when the $|m|$ is up to 8 times of l , the overall bandwidth will be up to 80%.

We compare the performance of our PLSC with that of the black-box hybrid scheme (Hybrid-PSEP, for short) of ref. [12]. Given the OTP (one time padding) encryption and PSEP suggested by the authors, the hybrid scheme of ref. [12] takes 4 hashes (one for OTP operation, two for commitment and one for Feistel transformation). Let $T_{se-hybrid-psep}$ and $T_{vd-hybrid-psep}$ be the Hybrid-PSEP's total computational cost of the signcryption and de-signcryption respectively, then

$$\begin{aligned} T_{se-hybrid-psep} &= T_r + 4T_h + 2T_{exp}, \\ T_{vd-hybrid-psep} &= 4T_h + 2T_{exp}. \end{aligned}$$

And its bandwidth and minimum length of ciphertext are the same as ours.

Table 1 summarizes the performance of our scheme as compared with the two previous works of Hybrid-PSEP and TBOS

We present some experimental results of our scheme. The experiments are implemented with Crypto++ 4.2 package on the platform of WINDOWS 2000 with SP 4, CPU of 2.60 GH and RAM of 512 MB. Table 2 shows the average CPU clocks consumed by each computing primitive for different lengths of messages (we keep the key length to be 1024 bits, and, for each message, we repeat the experiment 5 times. "MSGLEN" stands for the message length; "EN_G", "EN_H", "S_RSA" and "R_RSA" stand for the $\mathcal{G}(\cdot)$, $\mathcal{H}(\cdot)$, RSA computation of sender's signature key and RSA computation of receiver's encryption key respectively in signcryption; "R_DERSA", "S_DERSA", "DE_G", "DE_H" stand for the RSA computation of receiver's decryption key, the RSA computation of sender's verification key, $\mathcal{G}(\cdot)$ and $\mathcal{H}(\cdot)$ respectively in de-signcryption). Figure 2 illustrates the variation of time expenditure of each computing primitive in dealing with different lengths of messages.

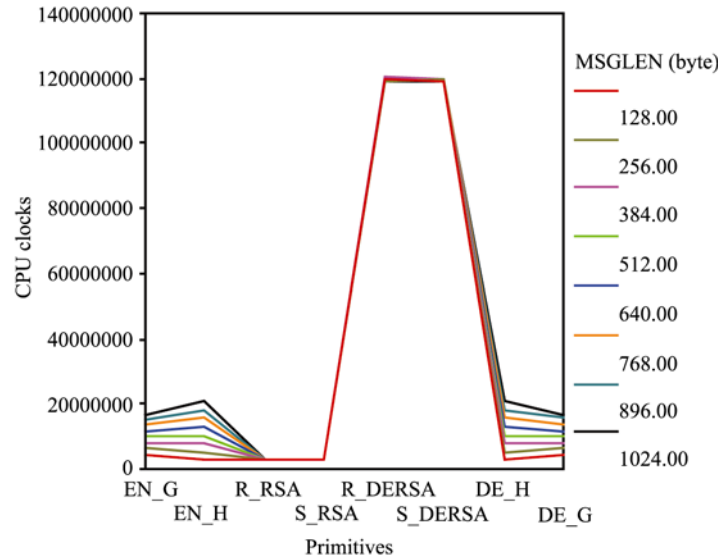
The experimental result shows that, for different lengths of messages, the time expenditure of hashing operation ascends in linearity, while the TDP's keeps at almost the same level. Though the time consumption of hashing varies directly as the message length, it is still negligible compared with the TDP's, and the overall time overhead is still dominated by the TDP operation.

Table 1 Performance comparison with prior schemes

Cost	Hybrid-PSEP	TBOS	PLSC
Computations in signcryption	$T_r + 4T_h + 2T_{exp}$	$T_r + 2T_h + 2T_{exp}$	$T_r + 2T_h + 2T_{exp}$
Computations in de-signcryption	$4T_h + 2T_{exp}$	$3T_h + 2.5T_{exp}$	$2T_h + 2T_{exp}$
Ciphertext length	$ m + 2 N_U $	$ N_U $	$ m + 2 N_U $

Table 2 CPU clocks consumed by each computing primitives for different length of messages

MSGLEN (byte)	EN_G (10^7)	EN_H (10^7)	R_RSA (10^7)	S_RSA (10^7)	R_DERSA (10^7)	S_DERSA (10^7)	DE_H (10^7)	DE_G (10^7)	Total (10^7)
128	0.46	0.28	0.29	0.29	11.94	11.97	0.28	0.46	25.97
256	0.64	0.51	0.29	0.29	11.96	11.93	0.52	0.64	26.78
384	0.81	0.80	0.29	0.29	11.97	11.98	0.81	0.81	27.76
512	0.98	1.04	0.29	0.29	11.99	11.93	1.04	0.99	28.55
640	1.16	1.28	0.29	0.29	11.97	11.93	1.30	1.16	29.38
768	1.34	1.56	0.29	0.29	11.96	11.96	1.56	1.34	30.30
896	1.51	1.80	0.29	0.29	11.95	11.97	1.80	1.53	31.14
1024	1.69	2.08	0.29	0.29	11.95	11.94	2.09	1.70	32.03

**Figure 2** Comparison of CPU clock consumption between different computing primitives for different lengths of messages.

5 Further discussions

One important point of the scheme captured in the pictures above is that the full public keys of both sender and recipient are clearly bound to the signcryption of m , along with the random string r . This prevents “identity fraud” attacks, and provides increased security for users who may publish multiple public keys.

Note that, in Definition 3 we, for the sake of conciseness, define the two TDPs used as “sign” and “encrypt” to compute data of the same length, however, all lengths can be chosen “appropriately” to support mismatched sender and recipient key lengths. This is important for legal reasons. For example, a sender might publish a certified public key using 1024-bit RSA, which is allowed to use in some countries for the purposes of signing, whereas encryption operations are limited to

512-bit modular by law. In this circumstance, if the sender wishes to signcrypt a message to a recipient with a 512-bit key, it needs not generate and sign a temporary 512-bit key; instead, the sender can simply apply the signcryption scheme directly using its 1024-bit sender's key and the 512-bit recipient's key.

If the desired application of signcryption requires “non-repudiation”, our scheme can be used to achieve this goal (barring disclosure of the sender's private key). A recipient who wishes to display the signature of the sender can simply decrypt c_1 , and provide it along with his public key and the sender's signature of c_3 as evidence, which can then be verified. This does not require the recipient to reveal his private key in order to prevent the sender from repudiating his message.

It may also be appropriate for settings where it is impractical to perform large block of messages (i.e. extremely low memory environments such as smart cards). To adapt to these settings, we suggest that hash functions used in our scheme should be constructed iteratively (e.g. from cipher block) like MGF used in PKCS#1 [18], and perform large messages in the “stream” manner: First, we choose the random string r , compute and output c_1 ; then we take the blocks of message one by one to generate each block of c_2 as well as the hash value g ; at last, we exert the inversion OTP operating on g to output c_3 .

6 Conclusion

In conclusion, we feel that our suggested signcryption scheme has proven to be a useful one. Protocol designers can easily use it to achieve extremely powerful security guarantees in a wide variety of settings. It is efficient, insider and forward secure, non-repudiation, and easily implemented. Our scheme also provides many small hidden benefits that will likely be found to use in time. For example, a user would possibly find that he could replace the TDPs in our scheme with a secure signature and an encryption scheme respectively with the only slight cost of performance degradation. Another, it is possible to distribute the portion of the output of $\mathcal{G}(\cdot)$ and the random string r that needs to be “signed” and “encrypted” over to any other machines, using any authenticated channel. The machine performing the “signing” or “encrypting” operation learns nothing about the ciphertext. Also, we can pre-compute the $f_R(\cdot)$ and $\mathcal{H}(\cdot)$ before the arrival of the message, which can even promote the performance dramatically.

As claimed above that the signcryption scheme we proposed is ideally suited to perform long messages, we would like to mention that if short messages ($|m| < l$) must be dealt with, the bandwidth would be getting terrible (as common hybrid methods do). We believe these drawbacks are extremely minor in light of the many advantages of our scheme.

Appendix A: Proof of Theorem 1

A.1 Proof of eq. (1)

We first establish bounds for $Adv_{SC, \mathcal{A}}^{ind-cca2}$ and prove eq. (1). Recall that f_S, f_S^{-1} represent the sender's public key and secret key respectively, and f_R, f_R^{-1} the keys of the receiver.

Consider an adversary \mathcal{A} against the IND-CCA2 security of the scheme. We show the bound on \mathcal{A} 's success probability by proceeding through a sequence of games G_0, \dots, G_4 , where G_0 is the unmodified IND-CCA2 attack game. We define the event S_i to be that \mathcal{A} is successful in G_i . Each

new game G_i will have a corresponding analysis bounding the difference between $\Pr[S_i]$ and $\Pr[S_{i-1}]$. The final bound on \mathcal{A} 's advantage in G_0 will follow from the success probability $\Pr[S_4]$, and the bound on $\Pr[S_0] - \Pr[S_4]$ which can be established from the relative bounds on the intermediate games. Throughout the following analysis we will use $c^* = c_1^* || c_2^* || \sigma^*$ to denote the challenge ciphertext corresponding to m_b (where b is the challenge bit) that is issued to \mathcal{A} in the IND-CCA2 attack game in response the challenge oracle query.

(i) Game G_1 . Let G_1 be the same as G_0 (the unmodified IND-CCA2 attack game), except that we replace the random oracles and the \mathcal{SE} and \mathcal{VD} oracles with the following simulators:

1) Random oracle simulator $\tilde{\mathcal{G}}$ and $\tilde{\mathcal{H}}$ for IND-CCA2 proof. For each query \mathcal{A} makes to \mathcal{G} , of the form $\mathcal{G}(m^i || r^i, id_S || id_R)$, the simulator $\tilde{\mathcal{G}}$ first checks to see if $\mathcal{G}(m^i || r^i, id_S || id_R)$ was previously defined. If so, the $\tilde{\mathcal{G}}$ replies with the previously defined value. Otherwise, the $\tilde{\mathcal{G}}$ chooses a random $\sigma^i \in \{0, 1\}^l$ and defines $\mathcal{G}(m^i || r^i, id_S || id_R)$ to be $f_S(\sigma^i)$. The simulator $\tilde{\mathcal{G}}$ then records the pair $(\sigma^i, m^i || r^i, id_S || id_R)$ in a lookup table for future reference (used in the \mathcal{VD} oracle simulation), and returns $f_S(\sigma^i)$.

Note that the distribution of values returned by this random oracle simulator is still uniformly random, so this simulation is perfectly "honest". However, it now takes time approximately T_f to compute before responding to each query. The space cost will be $|\sigma| + |m| + |r| + 2|id| = 2l + n + 2|id|$, where $|id|$ is the length of id_U in bits.

The simulator $\tilde{\mathcal{H}}$ for the random oracle \mathcal{H} works in the same manner with the only except that the value $f_R(r)$ needs not be computed. The space cost will be $|p| + |r| + 2|id| = n + l + 2|id|$.

2) De-signcryption oracle simulator $\tilde{\mathcal{VD}}$ for IND-CCA2 proof. When a query $\mathcal{VD}(c)$ is issued, the simulator $\tilde{\mathcal{VD}}$ parses c as $c_1 || c_2 || \sigma$. Then $\tilde{\mathcal{VD}}$ thinks of itself as the intended receiver and searches the lookup tables of \mathcal{G} and \mathcal{H} to find a matching pair $(\sigma, id_S || id_R)$ in a triple $(\sigma, m || r, id_S || id_R)$, and a matching pair $(r, id_S || id_R)$ in a triple $(p, r, id_S || id_R)$. If either of such triples does not exist, the $\tilde{\mathcal{VD}}$ returns \perp , otherwise, since $\mathcal{H}(r, id_S || id_R)$ has been queried already, it computes $m \leftarrow c_2 \oplus \mathcal{H}(r, id_S || id_R)$, $c_1' \leftarrow f_R(r)$. If $c_1 = c_1'$ then $\tilde{\mathcal{VD}}$ returns m , else returns \perp . We note that if the lookup succeeds, the $\tilde{\mathcal{VD}}$ will return the correct output. The time cost is about T_f .

3) Signcryption oracle simulator $\tilde{\mathcal{SE}}$ for IND-CCA2 proof. When a query $\mathcal{SE}(m)$ is issued, the simulator $\tilde{\mathcal{SE}}$ thinks of itself as the intended sender and properly computes the response as follows:

- Choose a random string $r \in \{0, 1\}^l$, and a random string $c_2 \in \{0, 1\}^n$, compute $c_1 = f_R(r)$, and $p = m \oplus c_2$. Define $\mathcal{H}(r, id_S || id_R)$ to be p . Fail if $\mathcal{H}(r, id_S || id_R)$ was already defined.
- Choose a random $\sigma \in \{0, 1\}^l$, and compute $y = f_S(\sigma)$.
- Define $\mathcal{G}(m || r, id_S || id_R)$ to be y . Fail if $\mathcal{G}(m || r, id_S || id_R)$ was already defined.
- Return $c_1 || c_2 || \sigma$.

Observe, if the procedure does not fail, $\tilde{\mathcal{SE}}$ returns a valid response to the \mathcal{SE} query, as $\sigma = f_S^{-1}(y)$. The time cost is about $2 \cdot T_f$.

This completes the definitions of the oracle simulators, which G_1 uses to replace the functionality of the corresponding oracles from G_0 . Provided that these oracle simulators do not fail, G_0 is

identical to G_1 , so $\Pr[S_0]-\Pr[S_1]$ is bounded by the probability that a failure occurs.

Denote by $Fial_{\mathcal{V}_D}$ the event that the simulator $\widetilde{\mathcal{V}_D}$ fails. $\widetilde{\mathcal{V}_D}$ can only fail by incorrectly returning \perp when queried on a valid ciphertext. That is, \mathcal{A} queried a valid ciphertext containing a r for which it did not issue a corresponding $\mathcal{H}(r, id_S//id_R)$ or a pair (m, r) for which it did not issue a corresponding $\mathcal{G}(m//r, id_S//id_R)$ oracle query. The first case means that the corresponding p will be random, the probability that such a random $p=c_2 \oplus m$ is at most 2^{-n} . The second case means that the corresponding g will be random, the probability that such a random $g=f_S(\sigma)$ is at most 2^{-l} . Thus the total probability that a failure of this type occurs after q_D queries to the de-signcryption oracle will be

$$\Pr[Fial_{\mathcal{V}_D}] \leq q_D(2^{-n} + 2^{-l}). \quad (A1)$$

Denote by $Fial_{SE}$ the event that the simulator \widetilde{SE} fails. The simulator \widetilde{SE} fails only if there is a ‘‘collision’’ between a freshly generated $(m//r, id_S//id_R)$ and one of the $(m'//r', id_S//id_R)$ for which $\mathcal{G}(m'//r', id_S//id_R)$ or $\mathcal{H}(r', id_S//id_R)$ have already been defined. We note there are at most $q_S + q_H$ and $q_S + q_G$ of such previously defined $\mathcal{H}(r', id_S//id_R)$ and $\mathcal{G}(m'//r', id_S//id_R)$ inputs respectively. If r were generated randomly and the intended users kept unchanged, the probability that it would collide with a previously defined value r' during one of the queries to the SE simulator would be at most $q_S \cdot (q_S + q_H) \cdot 2^{-l}$. In the same way, the probability of such collision about $m//r$ would be at most $q_S \cdot (q_S + q_G) \cdot 2^{-(n+l)}$. Thus the total probability that the simulator \widetilde{SE} fails after at most q_S queries will be

$$\Pr[Fial_{SE}] \leq q_S \cdot (q_S + q_H) \cdot 2^{-l} + q_S \cdot (q_S + q_G) \cdot 2^{-(n+l)} = q_S \cdot ((2^{-n} + 1)q_S + q_H + 2^{-n} \cdot q_G) \cdot 2^{-l}. \quad (A2)$$

Let $SimFial$ be the event that one of these simulators fails to return the correct response. Thus, by combining eq. (A1) with eq. (A2)

$$\begin{aligned} \Pr[SimFial] &\leq \Pr[Fial_{\mathcal{V}_D}] + \Pr[Fial_{SE}] \\ &\leq q_D(2^{-n} + 2^{-l}) + q_S \cdot ((2^{-n} + 1)q_S + q_H + 2^{-n} \cdot q_G) \cdot 2^{-l} \\ &\leq q_D \cdot 2^{-n} + (q_D + q_S \cdot ((2^{-n} + 1)q_S + q_H + 2^{-n} \cdot q_G)) \cdot 2^{-l}, \end{aligned} \quad (A3)$$

and we have now established the bound:

$$\Pr[S_0] - \Pr[S_1] \leq \Pr[SimFial]. \quad (A4)$$

(ii) Game G_2 . Game G_2 is G_1 modified to halt in the event that \mathcal{A} queries $\mathcal{H}(r^*, id_S//id_R)$ (where r^* denotes the corresponding random string of challenge ciphertext c^*) for any user id_S, id_R . We denote the event that game i halts prematurely in this fashion by $Halt_i$. Since G_2 is identical to G_1 unless $Halt_2$ occurs, we have

$$\Pr[S_1] - \Pr[S_2] \leq \Pr[Halt_2]. \quad (A5)$$

(iii) Game G_3 . Game G_3 is the same as G_2 , but modified so that the second component of the challenge ciphertext returned to \mathcal{A} , (i.e. c_2^*) is replaced by a random string. We note that the game halts if \mathcal{A} queries $\mathcal{H}(r^*, id_S//id_R)$ for any user id_S, id_R , so while the game is being played \mathcal{A} learns nothing about the value of $\mathcal{H}(r^*, id_S//id_R)$. In particular, this means that if we replace c_2^* by a random c'_2 , we can imagine that $\mathcal{H}(r^*, id_S//id_R)$ was defined ‘‘correctly’’ to be $m_b \oplus c'_2$ and thus G_3 has the same exact distribution as G_2 .

$$\Pr[S_3] = \Pr[S_2]. \quad (A6)$$

(iv) Game G_4 . Game G_4 is G_3 modified so that the entire challenge ciphertext c^* is replaced by a random string. That is, we replace c_1^* and σ^* by two random strings. We note that the challenge

ciphertext in G_3 does not provide \mathcal{A} with any information about $m_b \oplus p$, since c_2^* was replaced by a random string. Thus, it does not provide any information about the m_b , and for any random string σ' , we can think of $f_S(\sigma')$ as the “correct” value of $\mathcal{G}(m_b || r^*, id_S // id_R)$. Notice that, if the c_2^* and σ^* are replaced by random strings, the c_1^* cannot provide any information about the challenge message m_b . Thus G_4 has the same exact distribution as G_3 :

$$\Pr[S_4] = \Pr[S_3]. \quad (A7)$$

However, it is clear that since the challenge ciphertext was replaced by a random string, it no longer depends on the challenge bit b . Thus, \mathcal{A} must guess the bit correctly with probability $1/2$ in game G_4 when it plays to completion (i.e., when it does not halt):

$$\Pr[S_4] \leq 1/2 + \Pr[\text{Halt}_4]. \quad (A8)$$

Finally, we bound the probability that G_2 or G_4 halts by observing that the entire game can be run by a simulator with no knowledge of the secret trapdoor f_R^{-1} . Such a simulator can act as a reduction to the security of the underlying TDP. First, we observe the running of the game G_2 . The r^* is randomly chosen and the value $c^*_1 = f_R(r^*)$ is passed to the adversary \mathcal{A} for a portion of the challenge ciphertext, and the game will halt if the adversary issues a query of the form $\mathcal{H}(r^*, \cdot)$ such that $f_R(r^*) = c^*_1$. Thus, if the game halts, the adversary simply outputs this pre-image r^* , a successful inversion of the TDP. Thus

$$\Pr[\text{Halt}_2] \leq \varepsilon_{TDP}. \quad (A9)$$

And also

$$\Pr[\text{Halt}_4] \leq \varepsilon_{TDP}. \quad (A10)$$

Combining these bounds on $\Pr[\text{Halt}_2]$ and $\Pr[\text{Halt}_4]$ with eqs. (A3)–(A10) gives the desired result:

$$\begin{aligned} Adv_{SC, \mathcal{A}}^{ind-cca2}(t, q_H, q_G, q_S, q_D) &= 2 \cdot \Pr[S_0] - 1 \\ &\leq 2 \cdot (\Pr[\text{SimFial}] + \varepsilon_{TDP} + 1/2 + \varepsilon_{TDP}) - 1 \\ &= 2 \cdot (\Pr[\text{SimFial}] + 2 \cdot \varepsilon_{TDP}) \\ &\leq q_D \cdot 2^{-n+1} + (q_D + q_S \cdot ((2^{-n} + 1)q_S + q_H + 2^{-n} \cdot q_G)) \cdot 2^{-l+1} + 4 \cdot \varepsilon_{TDP}, \end{aligned} \quad (A11)$$

and eq. (1) holds.

A.2 Proof of eq. (2)

We then establish bounds for $Adv_{SC, \mathcal{B}}^{int-ctxt}$. To establish a bound on $Adv_{SC, \mathcal{B}}^{int-ctxt}$, we will reduce the adversary \mathcal{B} that breaks INT-CTXT security of the signcryption scheme to that breaks the security of the underlying TDP. The reduction will run a modified INT-CTXT attack game against \mathcal{B} , simulating the random oracles, a $\mathcal{V}\mathcal{D}$ oracle and a $\mathcal{S}\mathcal{E}$ oracle in similar fashion to the simulations introduced in game G_1 used in the $Adv_{SC, \mathcal{A}}^{ind-cca2}$ bound analysis. The reduction \mathcal{B} takes an inversion challenge y as input and finds a pre-image x such that $y = f_U(x)$.

1) Random oracle simulators $\tilde{\mathcal{G}}$ and $\tilde{\mathcal{H}}$ as well as signcryption oracle simulator $\tilde{\mathcal{S}\mathcal{E}}$ for INT-CTXT proof. These three oracle simulators are identical to the simulators from the IND-CCA2 proof.

2) De-signcryption oracle simulator $\tilde{\mathcal{V}\mathcal{D}}$ for INT-CTXT proof. This oracle simulator is constructed identically to the simulator from the IND-CCA2 proof. Notice that there is no challenge

ciphertext in this game, so the simulator only outputs \perp on ciphertexts for which $\mathcal{G}(m//r, id_S//id_R)$ or $\mathcal{H}(r, id_S//id_R)$ was not queried, or de-signcrypt legitimately returns \perp .

Let *SimFail* denote the event that one of the oracle simulations fails to return the correct response. The random oracle simulation is always correct, but the \mathcal{VD} oracle may fail exactly as in the failure case described in the IND-CCA proof. Similarly, the \mathcal{SE} oracle fails with the same probability as in the IND-CCA2 proof. The total probability of *SimFail* occurring is thus bounded as follows:

$$\begin{aligned} \Pr[\text{SimFail}] &\leq \Pr[\text{Fail}_{\mathcal{VD}}] + \Pr[\text{Fail}_{\mathcal{SE}}] \\ &\leq q_D \cdot 2^{-n} + (q_D + q_S \cdot ((2^{-n} + 1)q_S + q_H + 2^{-n} \cdot q_G)) \cdot 2^{-l}. \end{aligned} \quad (\text{A12})$$

Let *Bad* be the event that the new valid ciphertext forged by \mathcal{B} contains a pair (m, r) for which it never issue a corresponding $\mathcal{G}(m//r, id_S//id_R)$ or $\mathcal{H}(r, id_S//id_R)$ oracle query, which means that the corresponding g or p will be random. The probability that such a random g satisfying $g = f_S(\sigma)$ is at most 2^{-l} , and the probability that such a random p satisfying $p = c_2 \oplus m$ is at most 2^{-n} , so

$$\Pr[\text{Bad}] \leq 2^{-l} + 2^{-n}. \quad (\text{A13})$$

Notice that, when $\neg \text{Bad}$ occurs, both the corresponding $\mathcal{G}(m//r, id_S//id_R)$ and $\mathcal{H}(r, id_S//id_R)$ have been queried ever. In this case, if \mathcal{B} succeed in forging a new ciphertext $c'_1//c'_2//\sigma'$, then the corresponding σ' must be new. So the value $\mathcal{G}(m//r, id_S//id_R)$ is the pre-image of σ' , because that $f_S(\sigma') = \mathcal{G}(m//r, id_S//id_R)$. Thus

$$\Pr[\mathcal{B} \text{ success} \wedge \neg \text{SimFail} \wedge \neg \text{Bad}] \leq \varepsilon_{\text{TDP}}. \quad (\text{A14})$$

Combining eqs. (A12)–(A14), we get

$$\begin{aligned} \text{Adv}_{\mathcal{SCB}}^{\text{int-ctxt}}(t, q_H, q_G, q_S, q_D) &= \Pr[\mathcal{B} \text{ success}] \\ &= \Pr[\mathcal{B} \text{ success} \wedge \text{SimFail}] + \Pr[\mathcal{B} \text{ success} \wedge \neg \text{SimFail}] \\ &\leq \Pr[\text{SimFail}] + \Pr[\mathcal{B} \text{ success} \wedge \neg \text{SimFail}] \\ &= \Pr[\text{SimFail}] + \Pr[\mathcal{B} \text{ success} \wedge \neg \text{SimFail} \wedge \text{Bad}] \\ &\quad + \Pr[\mathcal{B} \text{ success} \wedge \neg \text{SimFail} \wedge \neg \text{Bad}] \\ &\leq \Pr[\text{SimFail}] + \Pr[\text{Bad}] + \Pr[\mathcal{B} \text{ success} \wedge \neg \text{SimFail} \wedge \neg \text{Bad}] \\ &\leq q_D \cdot 2^{-n} + (q_D + q_S \cdot ((2^{-n} + 1)q_S + q_H + 2^{-n} \cdot q_G)) \cdot 2^{-l} + 2^{-l} + 2^{-n} + \varepsilon_{\text{TDP}} \\ &= (q_D + 1) 2^{-n} + (q_D + q_S \cdot ((2^{-n} + 1)q_S + q_H + 2^{-n} \cdot q_G) + 1) \cdot 2^{-l} + \varepsilon_{\text{TDP}}, \end{aligned}$$

and eq. (2) holds.

A.3 Proof of eq. (3)

We evaluate the complexity of the above proof. The runtime for the simulations (and thus the reductions corresponding to each of the games in IND-CCA2 proof, as well as the reduction in the INT-CTXT proof) is $\mathcal{O}((q_G + 2 \cdot q_S + q_D)T_f)$ where T_f is the time to compute a single TDP operation, and the space required is $\mathcal{O}((n + l + 2|id|)(q_H + q_S) + (2l + n + 2|id|)(q_G + q_S))$ where l is the input length of the TDPs, n is the length of message, and $|id|$ the length of user's identification.

- 1 An J H, Dodis Y, Rabin T. On the security of joint signature and encryption. In: Knudsen L, ed. Advances in Cryptology—EUROCRYPT'02, LNCS Vol. 2332. Berlin: Springer-Verlag, 2002. 83–107. Available from <http://eprint.iacr.org/2002/046/>
- 2 Zheng Y. Digital signcrypton or how to achieve cost (signature & encryption) cost (signature) + cost (encryption). In: Kaliski B S, ed. Advances in Cryptology—CRYPTO'97, LNCS Vol. 1294. Berlin: Springer-Verlag, 1997. 165–179

- 3 Zheng Y, Imai H. Efficient signcryption schemes on elliptic curves. *Inf Proc Lett*, 1998, 68(6): 227–233 [\[DOI\]](#)
- 4 Petersen H, Michels M. Cryptanalysis and improvement of signcryption schemes. *IEEE Comput Dig Commun*, 1998, 145(2): 140–151
- 5 He W, Wu T. Cryptanalysis and improvement of petersen-michels signcryption schemes. *IEEE Comput Dig Commun*, 1999, 146(2): 123–124 [\[DOI\]](#)
- 6 Baek J, Steinfeld R, Zheng Y. Formal proofs for the security of signcryption. In: Naccache D, Pailler P, eds. 5th International Workshop on Practice and Theory in Public Key Cryptosystems PKC 2002, LNCS Vol. 2274. Berlin: Springer-Verlag, 2002. 80–98
- 7 Bellare M, Rogaway P. Optimal asymmetric encryption. In: Santis A D, ed. *Advances in Cryptology—EUROCRYPT 94*, LNCS Vol. 950. Berlin: Springer-Verlag, 1995. 92–111. Revised version available from <http://www-cse.ucsd.edu/users/mihir/>
- 8 Shoup V. OAEP reconsidered. In: Kilian J, ed. *Advances in Cryptology—CRYPTO 2001*, LNCS Vol. 2139. Berlin: Springer-Verlag, 2001. 240–259
- 9 Bellare M, Rogaway P. The exact security of digital signatures: How to sign with RSA and Rabin. In: Maurer U, ed. *Advances in Cryptology—EUROCRYPT 96*, LNCS Vol. 1070. Berlin: Springer-Verlag, 1996. 399–416. Revised version appears in <http://www-cse.ucsd.edu/users/mihir/papers/crypto-papers.html>
- 10 Fujisaki E, Okamoto T. Secure integration of asymmetric and symmetric encryption schemes. In: Wiener M, ed. *Advances in Cryptology—Proceedings of CRYPTO'99*, LNCS Vol. 1666. Berlin: Springer-Verlag, 1999. 537–554
- 11 Shoup V. A proposal for an ISO standard of public key encryption (version 2.1). *Cryptology ePrint Archive*, Report 2001/112. <http://eprint.iacr.org/2001/112>
- 12 Dodis Y, Freedman M J, Jarecki S, et al. Optimal signcryption from any trapdoor permutation. *Cryptology ePrint Archive*, Report 2004/20. <http://eprint.iacr.org/2004/20>
- 13 Mao W, Malone-Lee J. Two birds one stone: Signcryption using RSA. In: Joye M, ed. *Progress in Cryptology—CT-RSA 2003*, LNCS Vol. 2612. Berlin: Springer-Verlag, 2003. 210–224
- 14 Kobara K, Imai H. Oaep++: A very simple way to apply oaep to deterministic ow-cpa primitives. *Cryptology ePrint Archive*, Report 2002/130. <http://eprint.iacr.org/2002/130>
- 15 Luby M, Rackoff C. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J Comput*, 1988, 17(2): 373–386 [\[DOI\]](#)
- 16 Dodis Y, Freedman M J, Walfish S. Parallel signcryption with OAEP, PSS-R, and other Feistel paddings. *Cryptology ePrint Archive*, Report 2003/043. <http://eprint.iacr.org/2003/043>
- 17 An J H. Authenticated encryption in the public-key setting: Security notions and analyses. *Cryptology ePrint Archive*, Report 2001/079. <http://eprint.iacr.org/2001/079>
- 18 RSA Laboratories. PKCS #1 v2.1: RSA encryption standard, June 2002. Available from <http://www.rsa.com/rsalabs/pubs/PKCS/>
- 19 Rivest R L, Shamir A, Adlema L M. A method for obtaining digital signatures and public-key cryptosystems. *Commun ACM*, 1978, 21(2): 120–126