Research Article

Tarun Kumar Bansal*, Xavier Boyen and Josef Pieprzyk

Signcryption schemes with insider security in an ideal permutation model

https://doi.org/10.1515/jmc-2018-0006
Received February 19, 2018; revised March 19, 2019; accepted March 19, 2019

Abstract: Signcryption aims to provide both confidentiality and authentication of messages more efficiently than performing encryption and signing independently. The "Commit-then-Sign & Encrypt" (CtS&E) method allows to perform encryption and signing in parallel. Parallel execution of cryptographic algorithms decreases the computation time needed to signcrypt messages. CtS&E uses weaker cryptographic primitives in a generic way to achieve a strong security notion of signcryption. Various message pre-processing schemes, also known as message padding, have been used in signcryption as a commitment scheme in CtS&E. Due to its elegance and versatility, the sponge structure turns out to be a useful tool for designing new padding schemes such as SpAEP [T. K. Bansal, D. Chang and S. K. Sanadhya, Sponge based CCA2 secure asymmetric encryption for arbitrary length message, in: *Information Security and Privacy – ACISP 2015*, Lecture Notes in Comput. Sci. 9144, Springer, Berlin (2015), 93–106], while offering further avenues for optimization and parallelism in the context of signcryption. In this work, we design a generic and efficient signcryption scheme featuring parallel encryption and signature on top of a sponge-based message-padding underlying structure. Unlike other existing schemes, the proposed scheme also supports arbitrarily long messages. We prove the construction secure when instantiated from weakly secure asymmetric primitives such as a trapdoor one-way encryption and a universal unforgeable signature. With a careful analysis and simple tweaks, we demonstrate how different combinations of weakly secure probabilistic and deterministic encryption and signature schemes can be used to construct a strongly secure signcryption scheme, further broadening the choices of underlying primitives to cover essentially any combination thereof. To the best of our knowledge, this is the first signcryption scheme based on the sponge structure that also offers strong security using weakly secure underlying asymmetric primitives, even deterministic ones, along with the ability to handle long messages, efficiently.

Keywords: Signcryption, sponge structure, universal forgery, message padding, provable security

MSC 2010: 94A60, 68P25

Communicated by: Carlo Blundo

1 Introduction

The aim of signcryption is to provide both confidentiality and authentication of messages more efficiently than performing encryption and signing independently. The reduction of the computational cost makes signcryption more practical and it is a preferred option for e-commerce and e-mail applications, where both

***Corresponding author: Tarun Kumar Bansal,** ESY2, Robert Bosch Engineering and Business Solutions Private Limited (RBEI), Bangalore, India, e-mail: tkbansal17@gmail.com. http://orcid.org/0000-0003-1346-9206
Xavier Boyen, Faculty of Science and Engineering, Queensland University of Technology, Brisbane, Australia, e-mail: xavier.boyen@qut.edu.au
Josef Pieprzyk, Data61, CSIRO, Sydney, Australia; and Institute of Computer Science, Polish Academy of Sciences, Poland, e-mail: josef.pieprzyk@csiro.au

confidentiality and authentication are required. Zheng [38] introduced the signcryption notion in 1997. He proposes a signcryption solution that is based on the El-Gamal [25] encryption and signature, leaving the design of generic signcryption schemes as an open problem, which has since then received considerable attention.

The study of generic compositions of encryption and signature has been initiated by An, Dodis and Rabin [2]. They considered different methods for designing signcryption through a black-box composition of secure signature and public-key encryption. In particular, they showed that both "encrypt-then-sign" (EtS) and "sign-then-encrypt" (StE) lead to secure signcryption schemes. However, the parallel signcryption approach "sign-and-encrypt" (S&E) composition does not provide privacy since the signature may reveal information about the encrypted messages. They introduced an alternative generic method termed "commit-then-sign-and-encrypt" (CtS&E) that provides some security guarantee for S&E. Note that CtS&E compositions lead to parallel signcryption.

An, Dodis and Rabin [2] also define two types of security for signcryption, namely, outsider and insider security. The outsider security deals with an external adversary who knows the public keys of a sender and a receiver. The insider security model attacks are coming from the other party that participates in the communication. In other words, an insider adversary is either the sender who wants to compromise receiver confidentiality or the receiver who tries to defeat sender unforgeability. Since security against an insider adversary implies security against an outsider adversary, the former is preferred.

A different security model for signcryption, which has been adopted in a few early papers [2, 20], is the two-user setting. In this model, a single sender interacts with a single receiver. However, as pointed out by Dent [20], security in the two-user model does not imply security in the multi-user model, in which either several senders communicate with the same receiver or, alternatively, several receivers obtain messages from a single sender. Hence, to ensure a realistic security concept, a multi-user security model must be adopted. The strongest security definitions, which captures both insider confidentiality and unforgeability for the multi-user setting, have been defined in [29]. For an overview of different security models, see [21, 31]. A recent paper by Badertsche, Banfi and Maurer [3] also supports the need for an insider secure multi-user model for signcryption.

1.1 Background

In 2002, An, Dodis and Rabin [2] presented a methodology for parallel encryption and signing. A plaintext m is first transformed into a pair (c, d), where c is a commitment and d is a de-commitment. The value c reveals no information about m, while the pair (c, d) allows to recover m. Once the transformation $m \rightarrow (c, d)$ is done, the sender signs c and encrypts d in parallel using appropriate encryption and signature algorithms. On the receiver side, the signature on c is verified and d is recovered from its ciphertext. Both operations are executed in parallel. Finally, the plaintext m is reconstructed from (c, d). Parallel execution of cryptographic algorithms decreases the computation time needed to signcrypt a message. It is equal to the maximum of either the time required to encrypt or the time needed to sign. Minimum security requirements required from underlying encryption and signature algorithms are also discussed. In the two-user model, An, Dodis and Rabin [2] claim that to provide a generic chosen-ciphertext (IND-gCCA) secure and existentially unforgeable (UF-CMA) signcryption, it is enough to use any IND-CCA secure encryption, UF-CMA secure signature and a secure commitment under the CtS&E composition. The IND-gCCA security is weaker than IND-CCA.

The work by An, Dodis and Rabin [2] has instigated investigation into new ways to define signcryption in more generic ways. Note that early works present signcryption whose security depends on intractable problems such as discrete logarithm [38] and integer factoring [30, 36]. The authors of earlier works left an open question of designing signcryption under weaker security assumptions for encryption and signature schemes that do not relate to any specific intractability assumption. For example, the generic trapdoor one-wayness (OW) assumption is satisfied by the RSA encryption (when integer factorization is intractable) and the ElGamal encryption (when the computational Diffie–Hellman (CDH) problem is intractable). In this paper, we consider cryptographic primitives (encryption and signature), whose security assumptions are generic.

Parallel signcryption is further investigated by Pieprzyk and Pointcheval [33]. They proposed to use a (2, 2)-Shamir secret sharing (SSS) as a commitment scheme. A plaintext m is first split into two shares (s_1, s_2), where any single share reveals no information about m. The first share s_1 is used as a commitment and signed, while the second s_2 is encrypted. The authors of [33] proposed two version of their scheme. The first version, called generic parallel signcryption, provides IND-CCA and UF-CMA security for signcryption using any IND-CCA secure encryption and UF-CMA secure signature. This result is the same as the one obtained in [2]. The second version, called optimal parallel signcryption, applies an asymmetric padding OAEP [7] as commitment scheme. This signcryption algorithm provides both IND-CCA and UF-CMA security in the random oracle (RO) model assuming any deterministic OW encryption (such as basic RSA) and any weakly secure deterministic signature (non-universally forgeable). The authors discuss the security of their schemes for the insider security model in a multi-user setting [34].

Dodis et al. [22, 23] propose a different approach to perform parallel signcryption. In their approach, they use a Feistel probabilistic padding, which can be viewed as a generalization of other existing probabilistic paddings such as OAEP [7], OAEP+ [35], PSS-R [8], etc. The authors argue that their signcryption provides IND-CCA and strong existential unforgeability (sUF-CMA) security assuming trapdoor one-way permutations only.

Hybrid signcryption is an attractive approach in the design of signcryption schemes. It follows the idea of hybrid encryption discussed in many works [1, 5, 15, 17, 18, 24, 27, 28]. Hybrid encryption consists of an asymmetric *key encapsulation mechanism* (KEM) and a symmetric *data encapsulation mechanism* (DEM). The first formal treatment of security of signcryption has been done by Dent [19, 20]. Some other related works are [14, 16, 31, 37]. Converting a hybrid encryption scheme to hybrid signcryption turns out to be much trickier than it looks. The main difficulty is an increase in complexity of analysis that results from a more complex adversarial model. It is necessary to consider not only straightforward attacks against authenticity and confidentiality of messages but also more intricate issues such as distinction between outsider and insider attacks. Moreover, CtS&E-type compositions are always preferred as a base for constructing secure KEMs.

1.2 Limitation of existing schemes

A majority of signcryption schemes follow the sequential designs StE or EtS. Note that all schemes for hybrid signcryption with KEM/DEM [14, 16, 19, 20] follow the sequential design. The sequential design limits the efficiency of signcryption. This limitation can be lifted by using the CtS&E composition, which performs encryption and signing in parallel and independently from each other. Many signcryption schemes are built using some specific intractability assumptions (for example, intractability of discrete logarithm [4, 29, 38]). These constructions are not generic as the assumptions limit the choice of underlying encryption and signature schemes. Constructions for hybrid signcryption are generic, but they require stronger security properties from key and data encapsulation mechanisms. For example, a recent generic hybrid signcryption scheme given by Chiba et al. [16] requires an IND-CCA secure KEM, a one-time secure symmetric-key encryption, a one-time secure message authentication code and a strong existentially unforgeable signature scheme. These requirements are much stronger than those needed in already available non-hybrid schemes [33].

To the best of our knowledge, there is no hybrid signcryption that claims IND-CCA security and existential unforgeability using weak security properties like one-wayness and universal unforgeability. Most of the signcryption schemes require existential unforgeability for the underlying signature scheme, which is a stronger assumption than universal unforgeability. A common method used to build CtS&E-type scheme [22, 30, 33, 34] is an OAEP-type padding. The padding gives rise to some common limitations such as: (1) it restricts message space, (2) it works with deterministic one-way encryption and deterministic signature only and (3) it provides security in the random oracle (RO) model. Unavailability of different types of padding schemes limits the extension of work for the CtS&E composition. Table 1 gives a brief summary of generic signcryption schemes based on CtS&E.

Schemes	Model	Encryption	Signature	Message length	# of other functions	Signcryption
An, Dodis and Rabin [2]	No specific	IND-CCA	UF-CMA	Restricted	Commitment scheme	IND-gCCA/UF-CMA
Pieprzyk and Pointcheval [33]	Random oracle	OW-CPA	suUF-RMA	Restricted	3 hash, 1 secret share scheme	IND-CCA/sUF-CMA
Dodis et al. [22, 23]	Random oracle	OW-CPA	sUF-CMA	Restricted	1 hash, 1 commitment scheme	IND-CCA/sUF-CMA
				Unrestricted	1 hash, 1 commitment scheme, symmetric encryption	
Our result	Ideal permutation	OW-CPA	suUF-RMA	Unrestricted	1 SpongeWrap, 1 sponge function (≈ 2 hash)	IND-CCA/sUF-CMA
		OW-PCA	uUF-RMA			IND-CCA/UF-CMA

Table 1: Generic signcryption schemes based on CtS&E-type composition, where IND stands for indistinguishability, OW for one-wayness, CPA/CMA for chosen plaintext/message attack, CCA for chosen ciphertext attack, UF for existential unforgeability, uUF for universal unforgeability, suUF for strong uUF, RMA for random message attack, gCCA for generic CCA, OW-CPA for trapdoor one-way permutation and OW-PCA for one-wayness under plaintext-checking attack.

1.3 Motivation

A randomized padding, like OAEP, is a powerful tool, which converts weakly secure fixed trapdoor one-way functions into public-key encryption that is secure against strong adaptive chosen ciphertext attacks. The padding has been used in signcryption as a part of the commitment scheme in the CtS&E composition. It is known that CtS&E allows the use of weak cryptographic primitives in a generic way to achieve a strong security of signcryption. A good example of such composition is the results by Pieprzyk and Pointcheval [33, 34], which integrate any one-way encryption system (such as the basic RSA) with a weakly secure signature (non-universally forgeable signatures) into a strong chosen-ciphertext secure and existentially unforgeable signcryption in the RO model. The limitation of functionality, like message space restriction or type of encryption scheme, is inherited from the commitment or padding scheme used.

Recently, motivated by the OAEP design, Bansal, Chang and Sanadhya [6] proposed another type of padding called SpAEP. SpAEP is based on the sponge permutation structure, where permutation is considered as an ideal permutation, and the resulting sponge has no restriction on maximum message space. Unlike KEM-DEM, the SpAEP padding provides a pathway to combine symmetric and asymmetric primitives without a strict delineation. In brief, SpAEP uses a versatile sponge function and SpongeWrap [11, 12, 26] in pipelined fashion, and a portion of its output is used as input to the asymmetric encryption. The padding provides similar security guarantees as OAEP, but it is more efficient. The SpAEP padding can be used with trapdoor one-way permutations only. The sponge-based padding SpAEP [6] is versatile and has been used in a different security model for asymmetric encryption based on an ideal permutation. The padding scheme supports arbitrarily long messages, uses small domain permutations and applies “on the fly” encryption. Its running time is equivalent to a hash function.

Motivated by versatility of the sponge-based padding and by amplification of security properties (as demonstrated in [33, 34]), we would like to develop a generic signcryption scheme that is secure in the ideal permutation model. We intend to use weak asymmetric primitives such as trapdoor one-way encryption and universal unforgeable signature. The scheme is designed to support arbitrarily long messages. Experimental comparisons of the proposed scheme with existing generic signcryption schemes based on implementation is beyond the scope of this paper. However, a structural comparative analysis is provided in the next section.

1.4 Structural comparison

Generally, runtime performance of any signcryption is determined by processing time of asymmetric primitives, irrespective of underlying message-padding scheme, except perhaps for very long plaintexts (which many existing signcryption schemes do not even support). Therefore, structural efficiency improvement plays a secondary role in overall performance of signcryption, with the primary role being played by the ability to use weaker and faster asymmetric components. Nevertheless, simple and feature-rich message padding is always required to widen the applicability and usability of signcryption.

The proposed scheme uses only one SpongeWrap function and one sponge function. From the efficiency point of view, the proposed scheme is optimal since only a single call of SpongeWrap (≅ one hash function) is required before parallel encryption and signature. One call to the sponge function is required after asymmetric encryption for a small amount of data. The reverse process features the same kind of optimality. Similar optimality remains while processing arbitrarily long messages. Moreover, the entire message padding scheme is based on the iterative structure of a single forward permutation, which also saves implementation effort.

When compared to other generic schemes, Pieprzyk and Pointcheval [33] use a hash function, a secret sharing and OAEP (2 hash functions). A similar overhead is seen in the construction proposed by Dodis et al. [22, 23]. A simple practical generic signcryption scheme is proposed by Dodis et al. [22, 23]. The authors proposed a padding scheme called P-pad, which is equivalent to OAEP+ [35]. A detailed comparison of OAEP+ and sponge-based padding (SpAEP) is provided by Bansal, Chang and Sanadhya [6], which shows sponge-based padding schemes are more efficient and practical compared to OAEP-type padding schemes. In case of arbitrarily long messages, the scheme of Dodis et al. [22, 23] requires an additional symmetric encryption unlike our proposed scheme. These additional requirements of different functions with different input-output settings increase the implementation effort. Therefore, overall, our proposed scheme provides a simple, better and feature-rich message padding scheme for construction of generic signcryption scheme.

1.5 Contributions

In this paper, we make the following contributions.

(i) We present a signcryption scheme in the ideal permutation model using sponge structure. First we propose signcryption for messages of a fixed length. Then we show how to extend it for arbitrarily long messages. With careful analysis, we demonstrate how different combinations of weakly secure probabilistic/deterministic encryption and signature schemes can be used to build strongly secure generic signcryption. To the best of our knowledge, this is the first sponge-based signcryption. We also believe that the proposed signcryption is the first scheme, which allows different combination of weakly secure encryption and signature schemes to yield strongly secure signcryption that supports arbitrarily long messages.

(ii) The demands on component security are merely one-wayness for encryption and universal unforgeability for signature. These minimum security requirements are sufficient to achieve indistinguishability and existential unforgeability security against adaptive attacks. Such weak requirements were only fulfilled in [33, 34], but the scope of [33, 34] is limited to fixed message space and deterministic encryption and signatures.

(iii) Apart from encryption and signature primitives, our scheme requires an ideal permutation only. The iterative permutation model we use is based on the well-known iterative sponge structure. Note that, after the success of KECCAK [13] in the SHA-3 competition [39], the sponge structure is becoming more and more popular and can serve as a “Swiss army knife” in cryptography.

(iv) Flexibility of the sponge-based padding allows to scale the system from relatively short messages to long ones while preserving security properties. Besides, the complexity of the security analysis does not increase. Note that some extra redundant data is used in the proposed sponge padding that plays an important role in supporting long messages.

The sponge structure used for message padding resembles the padding proposed in [6] but differs in two aspects. First, some extra redundant data is used to allow the usage of sponge padding with a signature to provide both unforgeability and confidentiality. Second, while the padding in [6] applies for deterministic asymmetric encryption only, here we extend the sponge padding, so it also works with probabilistic asymmetric primitives.

Some properties are naturally inherited from the sponge structure. Signcryption offers an "on the fly" computation property during the signcryption and unsigncryption processes. An implementation does not need to use the inverse permutation, which saves implementation effort and memory.

Streaming. Our signcryption strategy enables unbuffered "on the fly" data processing (a.k.a. "streaming", "online", "single-pass" operation) during both the signcryption and unsigncryption processes. This is of significant interest when handling large messages, and one of the differentiating features of our scheme. For the avoidance of doubt, we note that single-pass unsigncryption necessarily requires that the recipient be able to discard an already decrypted stream that ends up failing authentication, with no persistent side-effect, for IND-CCA security. This operational limitation only applies to unsigncryption.

2 Preliminaries

Notations. In this work, we use $k \in \mathbb{N}$ as a security parameter, where $\mathbb{N}$ is the set of natural numbers. The symbol $|x|$ denotes the bit length of x, and $x \parallel y$ is a concatenation of x and y. If n is a positive integer, then the symbol $\{0,1\}^n$ denotes the set of n-bit strings. We also use $\{0,1\}^*$ to denote the set of binary strings of arbitrary length. $\lfloor X \rfloor_r$ represents first r bits of the string X, where $|X| \geq r$. Selecting a uniform and independently distributed variable x from a set I is denoted by $x \xleftarrow{\$} I$.

2.1 Ideal permutation

A permutation π is a bijective function on a finite domain D and range R, where $D = R$. An ideal permutation is a permutation chosen uniformly at random from all the available permutations. Let $D = R = \{0,1\}^b$, then $\pi \xleftarrow{\$} \mathrm{Perm}(D, D)$, where $\mathrm{Perm}(D, D)$ is the collection of all permutations on D. More precisely, $\pi\colon D \to R$ is a permutation if, for every $y \in R$, there is one and only one $x \in D$ such that $\pi(x) = y$.

2.2 Public-key encryption

Description. A public-key encryption scheme ENCRYPT is defined by the following three algorithms:

- the key generation algorithm $\mathsf{GenEnc}(1^k)$ that produces a pair (pk, sk) of public and private keys on input 1^k, where k is the security parameter,
- the encryption algorithm $\mathsf{Enc}_{\mathrm{pk}}(m; g) = c$ that outputs a ciphertext c for a message $m \in \mathcal{M}$ and a public key pk using random coins $g \in \mathtt{COINS}$ (the message and coin spaces $\mathcal{M}$ and $\mathtt{COINS}$ are uniquely determined by pk),
- the decryption algorithm $\mathsf{Dec}_{\mathrm{sk}}(c)$ that recovers a message m from a ciphertext c using a secret key sk.

We require that an asymmetric encryption scheme should satisfy the following correctness condition. For all $k \in \mathbb{N}$, for all (pk, sk) generated by $\mathsf{GenEnc}(1^k)$ and every $m \in \mathcal{M}$ and $g \in \mathtt{COINS}$, we always have $\mathsf{Dec}_{\mathrm{sk}}(\mathsf{Enc}_{\mathrm{pk}}(m; g)) = m$. We denote ℓ as a minimum input size and $\ell + \mathtt{co}_{\mathrm{pe}}$ as output size of Enc, where $\mathtt{co}_{\mathrm{pe}}$ is ciphertext overhead of Enc. The length of $g \in \mathtt{COINS}$ is denoted by λ, where $\lambda \geq k$. Enc is a deterministic encryption (trapdoor one-way permutation) if it does not require $g \in \mathtt{COINS}$ and $\mathtt{co}_{\mathrm{pe}} = 0$. If Enc depends upon $g \in \mathtt{COINS}$ and $\mathtt{co}_{\mathrm{pe}} > 0$, then Enc is a probabilistic encryption (trapdoor one-way function).

Security notions. The simplest security notion for public-key encryption is one-wayness (OW). This is to say that an adversary $\mathcal{C}$ cannot recover a plaintext m knowing a ciphertext c and a public key. We denote the

maximum probability of success that an adversary can invert the encryption of a random plaintext m in time t by $\mathrm{Succ}^{\mathrm{OW}}_{\mathcal{C},\mathrm{ENCRYPT}}$. OW is a minimal security requirement for public-key encryption. A variant of one-wayness is OW-PCA, which has been introduced in [32] for probabilistic encryption. For this notion, an adversary can additionally access a *plaintext checking oracle* ($\mathcal{O}^{\mathrm{PC}}$). The oracle $\mathcal{O}^{\mathrm{PC}}$ outputs 1 if a given (m, c) pair is a valid message/ciphertext pair for ENCRYPT; otherwise, it returns 0. As shown in [32], the ElGamal [25] encryption achieves OW-PCA under the GDH assumption. Clearly, for deterministic encryption, the OW and OW-PCA notions are the same.

A stronger security notion has also been defined. It is the so-called semantic security (a.k.a. indistinguishability of encryptions, IND). This is to say that a ciphertext should not leak any information about the encrypted message. More formally, knowing that a ciphertext is an encryption of one of two known messages, an adversary cannot guess the message with a non-negligible advantage. An adversary is seen as a 2-stage Turing machine $(\mathcal{A}_1, \mathcal{A}_2)$, and the advantage $\mathrm{Adv}^{\mathrm{IND}}_{\mathrm{ENCRYPT}}(\mathcal{A})$ is negligible for any adversary, where

$$\mathrm{Adv}^{\mathrm{IND}}_{\mathrm{ENCRYPT}}(\mathcal{A}) = 2 \times \Pr\left[\begin{array}{l}(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathsf{GenEnc}(1^k), (m_0, m_1, s) \leftarrow \mathcal{A}_1(\mathrm{pk}), \\ b \in \{0,1\},\ c = \mathsf{Enc}_{\mathrm{pk}}(m_b) : \mathcal{A}_2(m_0, m_1, s, c) = b\end{array}\right] - 1.$$

An adversary can try many different attacks. Knowing a public key, the adversary can encrypt any plaintext of its choice. This scenario is called the chosen-plaintext attack and denoted by CPA. Other attacks allow the adversary a restricted or unrestricted access to various oracles. The strongest attack allows the adversary to query the decryption oracle, which can be accessed adaptively in the chosen-ciphertext scenario (denoted as CCA). There is a restriction for queries – any query to the oracle should be different from the challenge ciphertext.

2.3 Signatures

Description. A digital signature SIGN consist of the following three algorithms:

- GenSign, the key generation algorithm, which, for a security parameter k, outputs a pair (pk, sk) of public and private keys,
- Sign, the signing algorithm, which takes a message M and the secret key sk and outputs a signature $\sigma = \mathsf{Sign}_{\mathrm{sk}}(M)$,
- Ver, the verification algorithm, which accepts a signature σ, a message M and a public key pk and returns a binary answer $\mathsf{Ver}_{\mathrm{pk}}(\sigma, M)$ (valid $\top$ or invalid $\perp$).

We assume that the signing algorithm takes an input of maximum ℓ_{sg} bits and that it generates a signature of length ℓ_σ.

Security notions. An adversary attempts to forge a signature. The probability of achieving this is assessed via the following game between a probabilistic polynomial time (PPT) adversary and a challenger.

(i) The challenger generates a key pair $(\mathrm{sk}, \mathrm{pk}) \leftarrow \mathsf{GenSign}(1^k)$.
(ii) The adversary runs $\mathcal{A}^{\mathcal{O}}(1^k, \mathrm{pk})$. They have access to an oracle $\mathcal{O}$ (which will be described below). The adversary terminates by outputting a message m^* and its signature σ^*.

In terms of resources, there are two types of attacks. The type of attack specifies the power that the adversary has in the attack.

- In a *no-message attack* (NMA), the oracle gives no response. This is equivalent to an attack model in which the adversary does not have access to the oracle $\mathcal{O}$. The adversary knows only the public key pk of the signer.
- In the second, a *known-message attack*, the adversary has access to a signature oracle providing a list of valid message/signature pairs in addition to knowledge of the public key of the signer. If this list contains random and uniformly chosen messages, then the attack is termed a random-message attack (RMA). If this list contains messages chosen by an adversary, the attack is termed a chosen-message attack (CMA). A chosen-message attack seeks to emulate the normal mode of use of a signature scheme, in which an adversary can observe signatures produced by a legitimate party, perhaps in some adversarial chosen way.

There are two ways, in which we can assess whether the adversary succeeds in forging a signature.

- Existential unforgeability (UF) – the adversary wins if it outputs a pair (m^*, σ^*), where $\mathsf{Ver}_{\mathrm{pk}}(m^*, \sigma^*) = \top$ and the adversary never queried the signature oracle with the message m^*.
- Strong existential unforgeability (sUF) – the adversary wins if it outputs a pair (m^*, σ^*), where the same conditions as for UF hold and, additionally, the adversary never received the response σ^*.

In case of a finite message space $\mathcal{M}$, we may consider a weaker security notion. For the success criteria, we may ask the adversary to produce a forged signature for a randomly chosen message $m^* \xleftarrow{\$} \mathcal{M}$. This leads us to a new game played by a probabilistic and polynomial-time adversary.

(i) The challenger generates a key pair $(\mathrm{sk}, \mathrm{pk}) \leftarrow \mathsf{GenSign}(1^k)$ and a message $m^* \xleftarrow{\$} \mathcal{M}$.
(ii) The adversary runs $\mathcal{A}^{\mathcal{O}}(1^k, \mathrm{pk}, m^*)$. It has access to an oracle $\mathcal{O}$. The adversary terminates by outputting a signature s^*.

We may define two success criteria for this security game.

- In the universal unforgeability (uUF) game, the adversary wins if $\mathsf{Ver}(\mathrm{pk}, m^*, \sigma^*) = \top$ and the adversary never queried the signature oracle with the message m^*.
- In the strong universal unforgeability (suUF) game, the adversary wins if $\mathsf{Ver}(\mathrm{pk}, m^*, \sigma^*) = \top$ and the adversary never queried the signature oracle with the message m^* nor received the response σ^*.

We say a signature is deterministic if signing a message multiple times results in the same signature. We say a signature is probabilistic if signing a message twice results in different signatures with overwhelming probability.

2.4 Signcryption: Joint encryption and signing

Description. A signcryption scheme SIGNCRYPT is defined by the following three algorithms:

- Gen, the key generation algorithm, which outputs a pair of keys (SDK, VEK) for a security parameter k, where SDK is the user's sign/decrypt key, which is kept secret, and VEK is the user's verify/encrypt key, which is made public,
- SignEnc, the encryption and signing algorithm, which, for a message M, the public key of the receiver VEK_R and the private key of the sender SDK_S, produces a signed ciphertext $Y = \mathsf{SignEnc}_{\mathrm{SDK}_S, \mathrm{VEK}_R}(M)$,
- VerDec, the decryption and verifying algorithm, which, for signed ciphertext Y, the private key SDK_R of the receiver and the public key VEK_S of the sender, recovers the message $M = \mathsf{VerDec}_{\mathrm{SDK}_R, \mathrm{VEK}_S}(Y)$. If this algorithm fails either to recover the message or to verify its authenticity, it returns $\perp$.

Security notions. We can combine classical security notions of signature and encryption to form a security notion of signcryption under adaptive attacks. Given access to public information $\mathrm{PUB} = (\mathrm{VEK}_S, \mathrm{VEK}_R)$ and oracle access to the functionalities of both sender S and receiver R, the adversary attempts to break

(i) authenticity (UF): coming up with a valid signed ciphertext of a new message, and thus provide an "existential forgery",
(ii) privacy (IND): breaking the "indistinguishability" of signed ciphertexts.

In the security analysis, the adversary may be one of S or R. So S may want to break the privacy, or R may want to break authenticity. If signcryption prevents existential forgeries and guarantees indistinguishability in the above attack scenarios (with chosen-message attacks CMA, or adaptive attacks AdA), we say the scheme is secure.

Definition 2.1. A signcryption scheme is *secure* if it achieves IND/UF under adaptive attacks.

3 Sponge-based padding

Description. Sponge-based padding consist two functions: *SpWrap* and *Sponge*. *SpWrap* and *Sponge* take some of their length parameters from ENCRYPT and SIGN used in SIGNCRYPT.

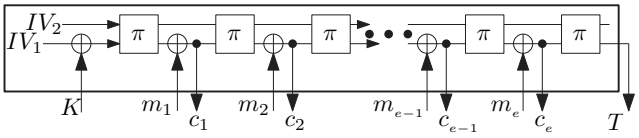

Figure 1: *SpWrap*$^{\pi}$, or simply *SpWrap*, and *Sponge* function.

SpWrap. This function is based on an iterated ideal permutation π: $\{0,1\}^{(b=r+c)} \to \{0,1\}^b$ with an initial value IV. It is a tuple of two algorithm *SpWrap.Enc*() and *SpWrap.Dec*().

On an input message M from message space Msg $\subset \{0,1\}^*$, *SpWrap.Enc*() gives the output $C \parallel T$ using a random K from the keyspace Key $\subset \{0,1\}^k$. *SpWrap.Enc*() takes the input message M, IV $= \mathrm{IV}_1 \parallel \mathrm{IV}_2$, K and some length parameters like k, r, ℓ_{sg}. The output of *SpWrap.Enc*() is $C \parallel T$, where $|C| > |M|$ and $|T| = k$. *SpWrap.Dec*() takes a ciphertext $C \parallel T$, IV $= \mathrm{IV}_1 \parallel \mathrm{IV}_2$, K and some length parameters like k, r, ℓ_{sg} as input. The output of *SpWrap.Dec*() is M or $\perp$.

SpWrap uses a structure similar to SpongeWrap [11], but its message padding is a little more specific than the general injective reversible padding used in SpongeWrap. After applying injective reversible padding to the input message, which is required for smooth functioning of the sponge structure, we specifically add a 0^r-bit block before the specific length ℓ_{sg}. This addition of an extra block is required during parallel signcryption to prevent some trivial forgery attack, which we will discuss later during the proof.

SpWrap.Enc$(K, M, \mathrm{IV}_1 \parallel \mathrm{IV}_2, r, k, \ell_{\mathrm{sg}})$

1 $x = \mathrm{IV}_1$; $w = \mathrm{IV}_2$
2 checkin$(M, r, k, \ell_{\mathrm{sg}}) = m_1 \parallel \cdots \parallel m_{(n+1)}$
3 $x = \mathrm{IV}_1 \oplus 0^{(r-k)} \parallel K$
4 **for** $i = 1, \ldots, n+1$ **do**
  $(x \parallel w) = \pi(x \parallel w)$; $x = x \oplus m_i$; $c_i = x$
5 $(x \parallel w) = \pi(x \parallel w)$; $T = \lfloor x \rfloor_k$
6 **return** $C \parallel T = c_1 \parallel c_2 \parallel \cdots \parallel c_{n+1} \parallel T$

SpWrap.Dec$(K, C \parallel T, \mathrm{IV}_1 \parallel \mathrm{IV}_2, r, k, \ell_{\mathrm{sg}})$

1 $c_1 \parallel c_2 \parallel \cdots \parallel c_{n+1} \parallel T = C \parallel T$, where each $|c_i| = r$
2 $x = \mathrm{IV}_1 \oplus 0^{(r-k)} \parallel K$; $w = \mathrm{IV}_2$
3 **for** $i = 1, \ldots, n+1$ **do**
  $(x \parallel w) = \pi(x \parallel w)$; $m_i = x \oplus c_i$; $x = c_i$
4 $(x \parallel w) = \pi(x \parallel w)$; $T' = \lfloor x \rfloor_k$
5 $X' = m_1 \parallel \cdots \parallel m_{n+1}$
6 **if** $T = T'$ **then**
  if there exists M such that $M =$ checkout$(X', r, k, \ell_{\mathrm{sg}})$ **then**
    return M
  else
    return $\perp$
  else
  return $\perp$

checkin$(M, r, k, \ell_{\mathrm{sg}})$

1 $X_1 \parallel X_2 = pad(M, r)$, where $|X_2| = \ell_{\mathrm{sg}} - r$
2 $X_1 \parallel 0^r \parallel X_2 = m_1 \parallel m_2 \parallel \cdots \parallel m_{n+1}$, where $|m_i| = r$ for $1 \le i \le (n+1)$ and there exists $m_i = 0^r$ such that $m_1 \parallel \cdots \parallel m_{i-1} = X_1$
3 **return** $m_1 \parallel m_2 \parallel \cdots \parallel m_{n+1}$

checkout$(X, r, k, \ell_{\mathrm{sg}})$

1 **if** there exist X_1, X_2 such that $X_1 \parallel 0^r \parallel X_2 = X$, where $|X_2| = \ell_{\mathrm{sg}} - r$ **then**
  $X' = X_1 \parallel X_2$
  else
  return $\perp$
2 **return** $unpad(X', r)$

$pad(x, r)$

$X = x \parallel 1 \parallel 0^{r-(|x|+1 \bmod r)-1} \parallel 1$
return X

$unpad(y, r)$

if there exists $x \neq \emptyset$ such that $x \parallel 1 \parallel 0^z \parallel 1 = y$, where $0 \le z \le r-1$, **then**
  return x
else
  return $\perp$

Sponge. This function works exactly like the sponge function in [26]. *Sponge* has fixed b-bit initial value IV, which is different from the IV of *SpWrap*. In *Sponge*, we take $\mathrm{IV} = \mathrm{IV}_1 \parallel \mathrm{IV}_3$, where $\mathrm{IV}_3 = \mathrm{IV}_2 \oplus 1$. *Sponge* takes $J \in \{0,1\}^*$ as input and outputs the k-bit tag value h. We define the *Sponge* function based on π as follows:

Sponge$(\mathrm{IV}_1 \parallel \mathrm{IV}_3, J)$

1 $x \parallel w = \mathrm{IV}_1 \parallel \mathrm{IV}_3$, where $|x| = r$
2 $j_1 \parallel j_2 \parallel \cdots \parallel j_n = pad(J, r)$, where $|j_i| = r$ for $1 \le i \le n$
3 **for** $i = 1, \ldots, n$ **do**
  $x = x \oplus j_i$; $x \parallel w = \pi(x \parallel w)$
4 **return** $\lfloor x \rfloor_k$

Properties. One useful property of *SpWrap* is its bijection. Considering a fixed IV for *SpWrap*, each query to *SpWrap.Enc*() has a fixed chain of internal variables because of the permutation π. Therefore, every query will have its unique set of state values. No two different queries can have a similar whole set of state bits. The first point of difference between two queries will create diversion in the set values because of the permutation π.

4 Parallel signcryption: SIGNCRYPT

In this section, we describe our proposal of parallel signcryption using sponge-based padding. To keep this scheme simple, we start with a restricted message space and a deterministic signature scheme. We remove these conditions in Section 5.

4.1 Description

Building blocks of parallel signcryption SIGNCRYPT are
- an encryption scheme ENCRYPT = (GenEnc, Enc, Dec),
- a signature scheme SIGN = (GenSig, Sign, Ver),
- a permutation $\pi\colon \{0,1\}^{(b=r+c)} \to \{0,1\}^b$ (assumed to behave like an ideal permutation),
- for k-bit security of parallel signcryption, π having sufficient $r > c > k$ such that it should provide at least k-bit security,
- assuming $\ell = n * r$ and $\ell_{\mathrm{sg}} = m * r$ for some positive integers $n, m > 0$,
- a public function *ID*, which maps the public key of any user A to a unique $\frac{r-k}{2}$-bit string in a compatible string format as ID_A, the communicating parties are denoted as sender S and receiver R,
- the length of a message M is $\ell + \ell_{\mathrm{sg}} - 2(k+1)$.

Algorithm (Key generation: $\mathsf{Gen}(1^k) = \mathsf{GenSig} \times \mathsf{GenEnc}(1^k)$). Sender S generates $(\mathrm{sk}^{\mathrm{sig}}, \mathrm{pk}^{\mathrm{sig}}) \leftarrow \mathsf{GenSig}_S(1^k)$ and receiver R generates $(\mathrm{sk}^{\mathrm{enc}}, \mathrm{pk}^{\mathrm{enc}}) \leftarrow \mathsf{GenEnc}_R(1^k)$. The sender keys are $(\mathrm{sk}_S, \mathrm{pk}_S) = (\mathrm{sk}^{\mathrm{sig}}, \mathrm{pk}^{\mathrm{sig}})$, and the receiver keys are $(\mathrm{sk}_R, \mathrm{pk}_R) = (\mathrm{sk}^{\mathrm{enc}}, \mathrm{pk}^{\mathrm{enc}})$. Accordingly, $\mathrm{SDK} = (\mathrm{sk}_S, \mathrm{sk}_R)$ and $\mathrm{VEK} = (\mathrm{pk}_S, \mathrm{pk}_R)$. Using the function *ID*, the unique identities of sender S and receiver R will be ID_S and ID_R, respectively.

Algorithm (Encrypt and sign: $\mathsf{SignEnc}_{\mathrm{SDK}_S, \mathrm{VEK}_R}(M)$).
(1) Compute $C \parallel T = SpWrap.Enc(K, M, \mathrm{IV}_1 \parallel \mathrm{IV}_2, r, k, \ell_{\mathrm{sg}})$, where $\mathrm{IV}_1 = \mathrm{ID}_S \parallel \mathrm{ID}_R$, $\mathrm{IV}_2 = 0^c$, $K \xleftarrow{\$} \{0,1\}^k$, $|K| = k$ and r is the input rate of π.
(2) Parse $C \parallel T$ into $S^1 \parallel S^2 \parallel T$, i.e., $C \parallel T = S^1 \parallel S^2 \parallel T$, where $|S^1| = \ell$, $|S^2| = \ell_{\mathrm{sg}}$.
(3) Calculate (in parallel) $Y_1 = \mathsf{Enc}_{\mathrm{pk}_R}(S^1)$, $\sigma = \mathsf{Sign}_{\mathrm{sk}_S}(S^2)$.
(4) Calculate $K_h = K \oplus Sponge(S^1 \parallel Y_1)$, $T_k = T \oplus K$.
(5) The final output $(K_h, Y_1, Y_2 = (S^2, \sigma), T_k)$ is sent to the receiver R.

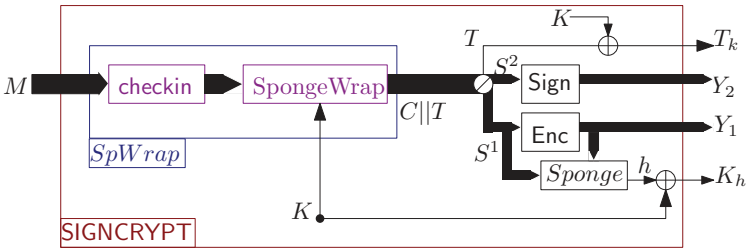

Figure 2: Signcryption scheme SIGNCRYPT.

Algorithm (Decrypt and verify: $\mathsf{VerDec}_{\mathrm{SDK}_R,\mathrm{VEK}_S}(K_h, Y_1, Y_2, T_k)$).

(1) Calculate (in parallel) $S^1 = \mathsf{Dec}_{\mathrm{sk}_R}(Y_1)$, $\top/\bot = \mathsf{Ver}_{\mathrm{pk}_S}(Y_2 = (S^2, \sigma))$. Ver returns either valid, $\top$, or $\bot$ if the signature is invalid. In case of returning $\bot$, the decryption and verify algorithm VerDec returns $\bot$ and stops.
(2) If Ver returns $\top$, then calculate $K = K_h \oplus \mathit{Sponge}(S^1 \parallel Y_1)$ and $T = T_k \oplus K$.
(3) Set $C = C^f \parallel C^e = S^1 \parallel S^2$, and set $\mathrm{IV}_1 = \mathrm{ID}_S \parallel \mathrm{ID}_R$, $\mathrm{IV}_2 = 0^c$.
(4) Compute $M' = \mathit{SpWrap.Dec}(K \parallel C \parallel T, \mathrm{IV}_1 \parallel \mathrm{IV}_2, r, k, \ell_{\mathrm{sg}})$. Return $M = M'$ if $M' \neq \bot$; else return $\bot$.

$\mathsf{SignEnc}_{\mathrm{sk}_S,\mathrm{pk}_R}(M)$

1 Initialization: $x = \mathrm{IV}_1 = 0^r$; $w = \mathrm{IV}_2 = 0^c$; $\mathrm{IV}_3 = \mathrm{IV}_2 \oplus 1$
2 Random key $K \xleftarrow{\$} \{0,1\}^k$
3 $\mathsf{checkin}(M, r, k, \ell) = m_1 \parallel \cdots \parallel m_{(n+1)}$
4 $x = \mathrm{ID}_S \parallel \mathrm{ID}_R \parallel K$
5 **for** $i = 1, \ldots, n+1$ **do**
  $(x \parallel w) = \pi(x \parallel w)$; $x = x \oplus m_i$; $c_i = x$
6 $(x \parallel w) = \pi(x \parallel w)$; $T = \lfloor x \rfloor_k$
7 $(S^1) \parallel (S^2) = (c_1 \parallel \cdots \parallel c_e) \parallel (c_{e+1} \parallel \cdots \parallel c_{n+1})$
8 $Y_1 = \mathsf{Enc}_{\mathrm{pk}_R}(S^1)$; $\sigma = \mathsf{Sign}_{\mathrm{sk}_S}(S^2)$
9 $\mathit{pad}(S^1 \parallel Y_1) = y_1 \parallel \cdots \parallel y_j$; $x = \mathrm{IV}_1$; $w = \mathrm{IV}_3$
10 **for** $i = 1, \ldots, j$ **do**
  $(x \parallel w) = \pi((x \oplus y_i) \parallel w)$
11 $K_h = \lfloor x \rfloor_k \oplus K$; $T_k = T \oplus K$
12 **return** $(K_h, Y_1, Y_2 = (S^2, \sigma), T_k)$

$\mathsf{VerDec}_{\mathrm{sk}_R,\mathrm{pk}_S}(K_h, Y_1, Y_2, T_k)$

1 Initialization: $\mathrm{IV}_1 = 0^r$; $\mathrm{IV}_2 = 0^c$; $\mathrm{IV}_3 = \mathrm{IV}_2 \oplus 1$
2 $S^1 = \mathsf{Dec}_{\mathrm{sk}_R}(Y_1)$; $x = \mathrm{IV}_1$; $w = \mathrm{IV}_3$
3 **if** $\mathsf{Ver}_{\mathrm{pk}_S}(Y_2 = (S^2, \sigma)) = \bot$ **then**
  return $\bot$
4 $(c_1 \parallel \cdots \parallel c_e) \parallel (c_{e+1} \parallel \cdots \parallel c_{n+1}) = (S^1) \parallel (S^2)$
5 $\mathit{pad}(S^1 \parallel Y_1) = y_1 \parallel \cdots \parallel y_j$
6 **for** $i = 1, \ldots, j$ **do**
  $(x \parallel w) = \pi((x \oplus y_i) \parallel w)$
7 $K = \lfloor x \rfloor_k \oplus K_h$; $T = T_k \oplus K$
8 $x = \mathrm{ID}_S \parallel \mathrm{ID}_R \parallel K$; $w = \mathrm{IV}_2$
9 **for** $i = 1, \ldots, n+1$ **do**
  $(x \parallel w) = \pi(x \parallel w)$; $m_i = x \oplus c_i$; $x = c_i$
10 $(x \parallel w) = \pi(x \parallel w)$; $T' = \lfloor x \rfloor_k$
11 $X' = m_1 \parallel \cdots \parallel m_{n+1}$
12 **if** $T = T'$ **then**
  if there exists M such that $M = \mathsf{checkout}(X', r, k, \ell)$ **then**
    return M
  else
    return $\bot$
 else
  return $\bot$

4.2 Security of parallel signcryption

Security of signcryption has two facets, namely, IND-CCA security and unforgeability under adaptive chosen message attack (UF-AdA). Before proceeding to the details of our proofs of each part individually, we provide a bird's eye view of each proof.

Theorem 4.1. *If the encryption scheme is* OW-PCA *and the signature scheme is deterministic* uUF-RMA, *then the parallel signcryption scheme described in Section 4.1 is* IND/UF-AdA *secure.*

Unforgeability

The following lemma can be derived from Theorem 4.1.

Lemma 4.2. *If there exists an adversary $\mathcal{A}$ against the* UF-AdA *security of the parallel signcryption scheme with advantage* $\mathrm{Adv}^{\text{UF-AdA}}_{\mathsf{SignEnc}}(k)$ *(whose running time is bounded by t and who makes at most $q_\pi^{\mathcal{A}}$ queries to the permutation $\pi\colon \{0,1\}^{b=r+c} \to \{0,1\}^b$ and q_{sc} queries to the signcryption oracle and q_{usc} queries to the unsigncryption oracle), then there exists an adversary $\mathcal{B}$ against the* uUF-RMA *security of the signature scheme with advantage* $\mathrm{Adv}^{\text{uUF-RMA}}_{\text{SIGN}}(k)$ *(whose running time is bounded by $t' \geq t + q_{\text{sc}}(\tau + O(1))$, where τ denotes the maximal running time of the encryption and signing algorithm) for which*

$$\mathrm{Adv}^{\text{UF-AdA}}_{\mathsf{SignEnc}}(k) \leq \mathrm{Adv}^{\text{uUF-RMA}}_{\text{SIGN}}(k) + \frac{(q_\pi - 1)q_\pi}{2^{b+1}} + \frac{q_\pi(q_\pi+1)}{2^c} + q_{\text{sc}} \cdot \frac{q_\pi^{\mathcal{A}}}{2^k} + \frac{q_{\text{usc}}}{2^k} + \frac{q_{\text{sc}}}{2^r},$$

where q_π is total number of π queries, including the queries by adversary, signcryption and unsigncryption oracles.

Proof sketch. We are dealing with the insider security model; the adversary has a target sender ID_S^* in mind and it knows the sender's public key pk_S^*. The adversary has access to the signcryption oracle under sk_S^*. In the multi-user setting, many receivers with different IDs are taken into consideration.

We make the subsequent changes in the permutation π such that π gives a permutation response for each new query but r bits out of the b-bit output are random. Likewise, c bits out of the b bit output are always different for new input. The bound of these changes will be $\frac{(q_\pi-1)q_\pi}{2^{b+1}} + \frac{q_\pi(q_\pi+1)}{2^c}$, where q_π is the number of total queries on π. In an abstract way, this bound includes collision over the b-bit and c-bit outputs of π.

We start making changes in the SignEnc oracle. We try to make the output of the SignEnc oracle random by using a random output of π. We use the message/signature pair list `Signlist` having q_H elements, where messages are chosen at random and signatures are calculated based on sk_{S^*}. Because we are working in the multi-user security model, SignEnc accepts different receiver's IDs along with M. Finally, SignEnc can respond with random output using a pre-computed `Signlist`, likewise independent of $\mathsf{Sign}_{\mathrm{sk}_{S^*}}$. The bound of changing the original response with a random response comes out to be $q_{\text{sc}} \cdot \frac{q_\pi^{\mathcal{A}}}{2^k}$. This bound captures the probability of guessing the randomness K used during the number of signcryption queries q_{sc}.

We modify the VerDec oracle such that we detect an existential forgery on VerDec and show a reduction to the universal forgery on Ver. Whenever we discuss a forgery, we consider $\mathrm{ID}_R = \mathrm{ID}_R^*$ in VerDec given by an adversary with target sign-ciphertext y^* and related pk_{R^*} and sk_{R^*} for target sender ID_{S^*}. For detecting a valid forgery, we cross-check the set I_{vd} consisting of the input/output of π during unsigncryption against a set $I_\pi^{\mathcal{A}}$ and I_π^{sc} consist of the input/output of π maintained by the adversary and the signcryption oracle, respectively. Let q_{usc} be the number of unsigncryption queries, and let q_{sc} be the number of signcryption queries. We show that if $I_{\text{vd}} \subset I_\pi^{\text{sc}}$, then this is not an existential forgery. We show that if ($I_{\text{vd}} \not\subset I_\pi^{\text{sc}}$ *and* $I_{\text{vd}} \not\subset I_\pi^{\mathcal{A}}$) or $I_{\text{vd}} \subseteq I_\pi^{\mathcal{A}}$, then the probability of having an existential forgery is negligible. The bound for these changes comes out to be $\frac{q_{\text{usc}}}{2^k} + \frac{q_{\text{sc}}}{2^r} + \mathrm{Adv}^{\text{uUF-RMA}}_{\text{SIGN}}(k)$. This bound captures the probability of producing a target collision on T or a target collision on the input of Ver or creating a signature on random input of Sign.

During the unforgeability proof, it is natural to assume that the encryption scheme is following trapdoor one-wayness and its correctness condition.

For a detailed proof, see Appendix A.

We can have the following corollaries from the proof of Lemma 4.2, which are also summarized in Table 2.

Corollary 4.3. *If the encryption scheme follows* OW-PCA *and the signature scheme is* uUF-RMA*, then the parallel signcryption scheme is* UF-AdA.

Corollary 4.3 is a direct implication of Lemma 4.2. This corollary includes both probabilistic and deterministic signature schemes and also encryption schemes.

Corollary 4.4 is a sub-class result of Corollary 4.3, where the deterministic signature scheme follows UF-AdA (or sUF-AdA). This corollary serves as a bridge for our next corollary, Corollary 4.5.

		Sign	
	Encrypt	uUF-RMA	suUF-RMA
Deterministic	**OW-CPA**	UF-AdA	sUF-AdA
Probabilistic	**OW-PCA**	UF-AdA	UF-AdA

Table 2: Unforgeability of SIGNCRYPT under different assumptions on Sign and Encrypt.

Corollary 4.4. *If the encryption scheme follows* OW-PCA*, and the signature scheme is* suUF-RMA*, then the parallel signcryption scheme is* UF-AdA.

Corollary 4.5. *If the encryption scheme is deterministic and follows one-wayness, and the signature scheme is* suUF-RMA*, then the parallel signcryption scheme is* sUF-AdA.

Corollaries 4.4 and 4.5 have a difference in achieved security because of the probabilistic and deterministic nature of the encryption scheme. This is mainly because the encryption scheme that follows OW-PCA includes some probabilistic asymmetric encryption schemes, which have a re-randomization problem. In re-randomization, for the same input to an asymmetric primitive, a different output value could be generated. In such a case and because of the insider security model, an adversary attacking the unforgeability of SIGNCRYPT can produce a different sign-ciphertext for the same input message, which is queried earlier. For example, for a query M, ID_R, the output is K_h, Y_1, Y_2, T_k for some K. Using insider knowledge and the probabilistic nature of asymmetric encryption, a new, valid output could be K'_h, Y'_1, Y_2, T_k for the same K and M, ID_R. Such a valid pair is allowed as part of forgery in sUF, but not in UF. Therefore, in Corollary 4.4, Sign follows suUF-RMA, but overall SIGNCRYPT follows only UF-AdA. If the encryption scheme is deterministic, then the above attack is not valid, and SIGNCRYPT can benefit from suUF-RMA. A summary of the above discussed corollary is shown in Table 2.

Privacy

The following lemma can be derived from Theorem 4.1.

Lemma 4.6. *Consider an adversary $\mathcal{A}$ against the* IND-CCA *security of the parallel signcryption scheme with advantage* $\mathrm{Adv}^{\text{IND-CCA}}_{\text{SIGNCRYPT}}(k)$ *whose running time is bounded by t and which makes at most q_π queries to the permutation $\pi\colon \{0,1\}^{b=r+c} \to \{0,1\}^b$ oracle and q_{usc} queries to the unsigncryption oracle. Then there exists an adversary $\mathcal{B}$ against the* OW-PCA *security of the public-key encryption scheme with advantage* $\mathrm{Adv}^{\text{OW-PCA}}_{\text{Encrypt}}(k)$ *and whose running time is bounded by $t' \le t + q_{\text{usc}}(\tau + O(1))$, where τ denotes the maximal running time of the decryption and verification algorithms, for which*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\text{SIGNCRYPT}}(k) \le \mathrm{Adv}^{\text{OW-PCA}}_{\text{Encrypt}}(k) + \frac{(q_\pi - 1)q_\pi}{2^{b+1}} + \frac{q_\pi(q_\pi + 1)}{2^c} + \frac{q_{\text{usc}}}{2^k} + \frac{q_\pi^{\mathcal{A}}}{2^k},$$

where q_π is the total number of π queries, including queries by the adversary ($q_\pi^{\mathcal{A}}$), signcryption and unsigncryption oracles, and $\lambda = k$.

Proof sketch. We are dealing with the insider security model in the multi-user setting; the adversary has a target receiver ID_R^* in mind. The adversary knows the receiver's public key pk_R and has access to the VerDec oracle under sk_R. Further, we assume that an adversary $\mathcal{A}$ observed q_{usc} queries to the VerDec oracle. Adversary $\mathcal{A}_1$ has also chosen a pair of messages M_0 and M_1 and a key pair $(\mathrm{sk}_S^*, \mathrm{pk}_S^*)$ for ID_S^*. It receives a ciphertext (y_1^*, y_2^*, y_3^*) under $(\mathrm{sk}_S^*, \mathrm{pk}_R^*)$ of either M_0 or M_1. The unknown message is denoted by M_d, where d is the bit that adversary $\mathcal{A}_2$ wishes to find out.

We make the subsequent changes in the permutation π such that π gives a permutation response for each new query, but r bits out of the b-bit output are random. Likewise, c bits out of the b-bit output are always different for new input. This part remains the same as for unforgeability.

	Sign	Encrypt OW-PCA
Deterministic	**uUF-RMA**	IND-CCA
	suUF-RMA	IND-CCA
Probabilistic	**uUF-RMA**	—
	suUF-RMA	IND-CCA

Table 3: Privacy of SIGNCRYPT under different combinations of Sign and Encrypt.

We modify the unsigncryption oracle such that it nullifies those queries to the unsigncryption oracle about which the adversary does not know an answer in advance with the help of the π query and which can be simulated without using the private key of the receiver sk_{R^*}. If $I_\pi^{\mathrm{vd}} \notin I_\pi^{\mathcal{A}}$, then the probability that the adversary can get an answer from the unsigncryption oracle is bounded by $\frac{q_{\mathrm{usc}}}{2^k}$, which includes target collision on T for the number of unsigncryption queries q_{usc}. Unlike unforgeability, the adversary is allowed to generate a valid signcryptext, but only those will be valid about which the adversary already knows the answer.

We modify the signcryption oracle using the random response of π. This will lead to simulating the signcryption oracle returning a random response. This change will be bounded by the probability of guessing the randomness K used by an adversary or the advantage of an OW-PCA adversary breaking the one-wayness (OW).

The privacy proof of the scheme depends upon the probabilistic or deterministic nature of the underlying signature scheme. During the proof, we assume that the signature scheme is deterministic and follows the correctness condition. In subsequent sections, we show how we can remove this assumption on the signature scheme.

For a detailed proof, see Appendix B. After the proof of Lemma 4.6, we can have the following corollaries.

Corollary 4.7. *If the encryption scheme is* OW-PCA *and the signature scheme is deterministic, then the parallel signcryption scheme is* IND-CCA.

This corollary follows directly from Lemma 4.6.

Corollary 4.8. *If the encryption scheme is deterministic* OW-CPA *and the signature scheme is deterministic, then the parallel signcryption scheme is* IND-CCA.

This corollary follows a sub-class result of Corollary 4.7, where the deterministic OW-CPA secure encryption scheme also follows OW-PCA.

Next, Corollary 4.9 is another representation of Corollaries 4.7 and 4.8, where we say only suUF-RMA signature schemes are valid for security because a deterministic uUF-RMA secure scheme also follows suUF-RMA.

Corollary 4.9. *If the encryption scheme is deterministic* OW-CPA *and the signature scheme* suUF-RMA, *then the parallel signcryption scheme is* IND-CCA.

Corollaries 4.4 and 4.7 together give Theorem 4.1. Corollaries 4.5 and 4.9 together give the following theorem, Theorem 4.10. A summary of the corollaries related to the privacy proof of SIGNCRYPT is shown in Table 3. A gap in the results, where probabilistic Sign following uUF-RMA does not provide security to SIGNCRYPT will be addressed in the next section.

Theorem 4.10. *If the encryption scheme is deterministic* OW-CPA *and the signature scheme is* suUF-RMA, *then the parallel signcryption scheme is* IND/sUF-AdA *secure*.

The proof of this theorem exactly follows the proof of Theorem 4.1, except that we now assume that Sign is suUF-RMA secure and Encrypt is also deterministic OW-CPA.

5 Extension of parallel signcryption

In Section 4, we saw two limitations of SIGNCRYPT. First, it does not support probabilistic Sign, where the same input can give two or more different signatures. Second, there is a restriction on the maximum message length. In this section, we discuss how to extend the usage of the parallel signcryption SIGNCRYPT in case of probabilistic Sign and in case of arbitrarily long messages.

5.1 Using probabilistic Sign

Probabilistic Signis not supported in the proposed scheme because we assumed Sign is deterministic and, for the same input, two different signatures are not considered. In cases where a probabilistic Sign scheme needs to be used, IND-CCA security of SIGNCRYPT will no longer be valid under the proposed scenario because now an insider adversary can simply produce another signature σ on S^{2*} of the challenged signed ciphertext and submit $K_h^*, Y_1^*, Y_2 = (S^{2*}, \sigma), T_k^*$ to VerDec. This will lead to knowing bit d of M_d with probability 1 without violating the IND-CCA experiment. This attack case can be handled easily in two ways.

Solution 1. Relaxing IND-CCA experiment to IND-gCCA [2]: Consider the challenged signed ciphertext $K_h^*, Y_1^*, Y_2 = (S^{2*}, \sigma^*), T_k^*$ as two parts. The first one is the ciphertext $(K_h^*, Y_1^*, S^{2*}, T_k^*)$, and the second one is the signature σ^*. Imposing a restriction on the adversary attacking the IND-CCA security, not only can the challenged signed ciphertext not be queried to the decryption oracle, but those queries that result in the same as the challenged ciphertext $K_h^*, Y_1^*, S^{2*}, T_k^*$ are also prohibited. A query to VerDec having the challenged ciphertext $K_h^*, Y_1^*, S^{2*}, T_k^*$ could be easily determined by using the public key of the sender as verification key.

This change in the IND-CCA experiment is similar to IND-gCCA proposed in [2]. An, Dodis and Rabin [2] proposed this IND-gCCA notion specifically for signcryption in a more formal way to avoid the trivial attack discussed above. By following the IND-gCCA security experiment in [2], we can propose another corollary from Lemma 4.6.

Corollary 5.1. *If the encryption scheme is* OW-PCA *and the signature scheme is unforgeable, then the parallel signcryption scheme is* IND-gCCA.

This corollary can be combined with corollaries from Lemma 4.2 and different, new results can be achieved.

Solution 2. Include σ also as part of the input in *Sponge*. This inclusion of σ in *Sponge* will bind σ to a particular K, S^2 like in the case of Y_1. Now the above discussed attack will not work because a different σ will lead to a different K. This change is more simple compared to the IND-gCCA security notion requirement. This change is initially not included in the proposed scheme with the intention to keep the proof simple and straight. Inclusion and reason of this proposed change helps in understanding about IND-gCCA and (Y_1, σ) as input to *Sponge*.

5.2 Arbitrarily long messages

An arbitrarily long message can be supported in SIGNCRYPT without any major structure modification. Earlier, $S^1 \parallel S^2 \parallel T = C \parallel T$ when $|C \parallel T| = \ell + \ell_{sg} + k$. If $|C \parallel T| > \ell + \ell_{sg} + k$, then $S^1 \parallel C^e \parallel S^2 \parallel T = C \parallel T$, where $|S^1| = \ell$, $|S^2| = \ell_{sg}$, and the final output of SIGNCRYPT is $(K_h, Y_1, C^e, Y_2 = (S^2, \sigma), T_k)$.

Caution. It is essential that if $|C \parallel T| > \ell + \ell_{sg} + k$, then $S^1 \parallel C^e \parallel S^2 \parallel T = C \parallel T$, *not* $S^1 \parallel S^2 \parallel C^e \parallel T = C \parallel T$, where S^1 is the input of Enc and S^2 is the input of Sign. This requirement to perform signing on the last part of the data arises in signcryption to prevent a trivial forgery attack by an insider adversary. In case where Sign is performed on data subsequent to Enc data, like $S^1 \parallel S^2 \parallel C^e \parallel T = C \parallel T$, then the adversary can replace C^e and accordingly T using $\pi^{\mathcal{A}}$, sk and pk of Enc. This modification will lead to a trivial forgery.

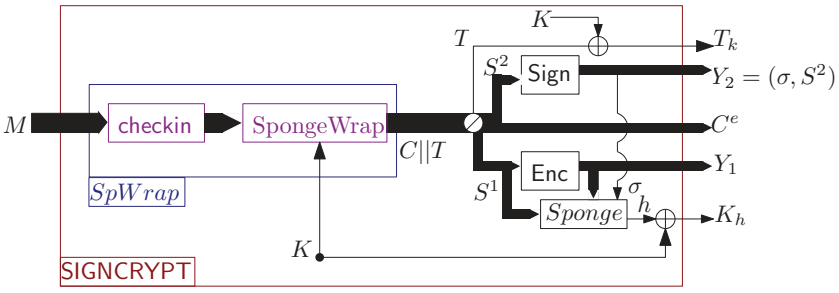

Figure 3: Signcryption scheme SIGNCRYPT^G. The input message is passed to *SpWrap*, which uses checkin and the SpongeWrap function, along with random K. *SpWrap* outputs $C \parallel T$, and $C \parallel T$ further splits into $S^1 \parallel C^e \parallel S^2 \parallel T$. The asymmetric encryption scheme Enc takes S^1 as input and outputs Y_1. The signature scheme Sign takes S^2 as input and outputs Y_2, which consists of (S^2, σ), where σ is the signature. *Sponge* takes Y_1, S^1 and σ as input and outputs h, which further gets xored with K to produce K_h. The final output will be K_h, Y_1, C^e, Y_2, T_k, where $T_k = T \oplus K$.

With this proposed change from solution 2 and support of long messages, we call SIGNCRYPT^G a generic version of SIGNCRYPT. A graphical representation of generic signcryption is shown in Figure 3.

Theorem 4.1 can be modified for SIGNCRYPT^G as follows:

Theorem 5.2. *If the encryption scheme is* OW-PCA *and the signature scheme is* (uUF,suUF)-RMA*, then the parallel signcryption* SIGNCRYPT^G *scheme is* IND-CCA/(UF,sUF)-AdA *secure.*

Proof sketch. If we follow the proof of Lemma 4.2, after game G5, we can clearly see that the output of π is random. Following random π, the output h of *Sponge* is also random. Even if the adversary tries to use another σ for the same S^2, this will result in a change of h that leads to random K and T_k, and the adversary needs to produce a target collision over that T or K. This case is already included in the proof when $I_{vd} \not\subset I_{sc}$ and $I_{vd} \not\subset I_\pi^{\mathcal{A}}$.

For IND-CCA security of SIGNCRYPT^G, we can follow the proof of Lemma 4.6 including extra cases when ENCRYPT and SIGN are probabilistic. In order to get information about M_d, now the adversary tries to produce different Y_1' for the same S^{1*} or different σ for the same S^{2*}. Either of these cases will change the value of K^*, which reduces the problem again of having a collision on T or having knowledge of S^1. This results in the same bound on IND-CCA2 as for SIGNCRYPT.

Therefore, regarding IND-CCA of SIGNCRYPT^G, the addition of *Sponge* is a dummy operation compared to SIGNCRYPT for outputting T, but its usage protects σ of SIGN and outputs Y_1 of ENCRYPT by making them dependent on a particular K. This dependency provides IND-CCA security for SIGNCRYPT^G in a similar way for SIGNCRYPT.

6 Conclusion

The combination of an encryption and a signature scheme yields a signcryption scheme. The extra burden of satisfying both privacy and unforgeability against insider adversaries increases the complexity of proving that the system is secure and efficient. This complexity brings limitations on the signcryption scheme in terms of the needed security assumptions, security achievement and efficiency to balance each other. Message pre-processing is found to be an attractive way to build a secure and efficient signcryption scheme. These message pre-processing techniques are found to be inflexible, which disallows their improvement in different scenarios like long message length, different types of underlying encryption and signature schemes, insider security, efficient computation in parallel, etc.

The versatile nature of the sponge structure enable us to modify message pre-processing efficiently. This efficient message pre-processing helps us to build a secure signcryption scheme achieving a higher security level using a weakly secure encryption and signature scheme. We also found that the probabilistic and deterministic nature of the signature scheme plays an important role in the privacy of the signcryption scheme, but the same is not true for unforgeability with respect to the encryption scheme. At the end, we were able

to find a signcryption scheme that can perform efficiently without compromising its security. The proposed scheme is highly customizable as it allows to use weakly secure schemes and different types of the underlying encryption and signature schemes.

A Proof of Lemma 4.2

Proof. We consider an experiment similar to UF-AdA as described in Section 2.3. We follow the subsequent experiment for UF-AdA security of SIGNCRYPT against adversary $\mathcal{A}$.

Experiment. $\mathrm{Exp}^{\text{UF-AdA}}_{\text{SIGNCRYPT},\mathcal{A}}(k)$

1 $(\mathrm{sk}_{S^*}, \mathrm{pk}_{S^*}) \leftarrow \mathsf{GenSig}_{S^*}(1^k)$
2 $(y^*, \mathrm{ID}_{R^*}) \leftarrow \mathcal{A}^{\mathsf{SignEnc}_{\mathrm{sk}_{S^*}}(\cdot,\cdot),\pi(\cdot)}(\mathrm{pk}_{S^*})$
3 Map pk_{R^*} using ID_{R^*}, where $(\mathrm{sk}_{R*}, \mathrm{pk}_{R*}) \leftarrow \mathsf{GenEnc}_{R^*}(1^k)$
4 $M^* \leftarrow \mathsf{VerDec}(\mathrm{pk}_{R^*}, \mathrm{sk}_{S^*}, y^*)$
5 **if** $M^* \neq \perp$ and the query (M^*, ID_{R^*}) is never made to the $\mathsf{SignEnc}_{\mathrm{sk}_{S^*}}(\cdot,\cdot)$ oracle **then**
  return 1
else
  return 0

The advantage of adversary $\mathcal{A}$ is given by

$$\mathrm{Adv}^{\text{UF-AdA}}_{\text{SIGNCRYPT}}(k) = \Pr[\mathrm{Exp}^{\text{UF-AdA}}_{\text{SIGNCRYPT},\mathcal{A}}(k) = 1].$$

We use a game-based proof framework [10]. We are dealing with the insider security model; the adversary has a target sender ID^*_S in mind and it knows the sender's public key pk^*_S. The adversary has access to the signcryption oracle under sk^*_S. We denote the winning event of forging a signcryptext in game i by G_i.

Game G0 represents the original signcryption game for UF-AdA. The adversary issues q_{sc} queries on the signcryption oracle specifying the receiver ID_R in each query using ID^*_S. Adversary $\mathcal{A}$'s target is to give a target ID^*_R and a signed ciphertext $(K^*_h, Y^*_1, Y^*_2 = (S^{2*}, \sigma^*), T^*_k)$ such that $\mathsf{VerDec}_{\mathrm{sk}^*_R,\mathrm{pk}^*_S}(K^*_h, Y^*_1, Y^*_2, T^*_k) = M^* \neq \perp$, where (M^*, ID^*_R) should not be queried by $\mathcal{A}$ to $\mathsf{SignEnc}$. Adversary $\mathcal{A}$ might ask (M, ID^*_R) or (M^*, ID_R) to $\mathsf{SignEnc}$. Therefore,

$$\Pr[G_0] = \Pr[\mathrm{Exp}^{\text{UF-AdA}}_{\text{G0},\mathcal{A}}(k) = 1] = \Pr[\mathrm{Exp}^{\text{UF-AdA}}_{\text{SIGNCRYPT},\mathcal{A}}(k) = 1].$$

From G0 to G4, we make successive changes in the permutation π. The modified π gives a permutation response for each new query such that r bits out of the b-bit output are random. Likewise, c bits out of the b-bit output are always different for new input. This helps us to exploit the permutation property of *Sponge* and make an output C deterministic for a specific input K, M and IV. Any change in either of the four values (C, M, K, IV) will make at least one value random. Here "any change" implies, while establishing a relation between (C, M, K, IV), if any input/output pair of π is not defined already, then essentially one of the parts is new or randomly generated.

Games G1 and G2. We start making changes in the permutation π. In G1, we take the response of π randomly and differently from the previous responses using the set I_π. In G2, π queries are simulated as a random function, that is, for every new input, the output is random, which does not need to be different. So, in G2, π gives a random response without cross-checking it in the previous input/output response list I_π. G1 and G2 remain identical until the output of the π query collides with any of the previous outputs. This collision is denoted as the event *bad*. The probability that a random response chosen as the output of π will collide with any previous response is $\frac{(q_\pi-1)q_\pi}{2^{b+1}}$, where q_π is the total number of queries on π (and π^{-1}), either from oracle calls by a different oracle or by the adversary $\mathcal{A}$. Therefore, $|\Pr[G_2] - \Pr[G_1]| \leq \frac{(q_\pi-1)q_\pi}{2^{b+1}}$.

Game G0.

Initialization: $I_\pi = I_\pi^{\mathcal{A}} = \emptyset$; $\mathrm{IV}_1 = 0^r$; $\mathrm{IV}_2 = 0^c$; $\mathrm{IV}_3 = \mathrm{IV}_2 \oplus 1$; $(\mathrm{sk}_R, \mathrm{pk}_R) \leftarrow \mathsf{GenEnc}(1^k)$, $\mathrm{pk}_S^*, \mathrm{ID}_S^*$; $|\mathrm{ID}| \in \{0,1\}^{(r-k)/2}$

`Signlist`: $\{(S_i, \sigma_i) : \sigma_i = \mathsf{Sign}_{\mathrm{sk}_S^*}(S_i)$ for all $1 \le i \le q_H$ and each S_i chosen randomly$\}$

On SignEnc-query M, ID_R

1 $K \xleftarrow{\$} \{0,1\}^k$; $x = \mathrm{IV}_1$; $w = \mathrm{IV}_2$
2 $\mathsf{checkin}(M, r, k, \ell_{\mathrm{sg}}) = m_1 \| \cdots \| m_{(n+1)}$
3 $x = \mathrm{ID}_S^* \| \mathrm{ID}_R \| K$
4 **for** $i = 1, \ldots, n+1$ **do**
  $(x \| w) = \pi(x \| w)$; $x = x \oplus m_i$; $c_i = x$
5 $(x \| w) = \pi(x \| w)$; $T = \lfloor x \rfloor_k$
6 $S^1 \| S^2 \| T = c_1 \| \cdots \| c_e \| c_{e+1} \| \cdots \| c_{n+1} \| T$
7 $Y_1 = \mathsf{Enc}_{\mathrm{pk}_R}(S^1)$; $\sigma = \mathsf{Sign}_{\mathrm{sk}_S^*}(S^2)$
8 $pad(S^1 \| Y_1) = y_1 \| \cdots \| y_j$; $x = \mathrm{IV}_1$; $w = \mathrm{IV}_3$
9 **for** $i = 1, \ldots, j$ **do**
  $(x \| w) = \pi(x \oplus y_i \| w)$
10 $K_h = \lfloor x \rfloor_k \oplus K$; $T_k = T \oplus K$
11 **return** $(K_h, Y_1, Y_2 = (S^2, \sigma), T_k)$

On VerDec-query (K_h, Y_1, Y_2, T_k)

1 $S^1 = \mathsf{Dec}_{\mathrm{sk}_R}(Y_1)$; $x = \mathrm{IV}_1$, $w = \mathrm{IV}_3$
2 **if** $\mathsf{Ver}_{\mathrm{pk}_S^*}(Y_2 = (S^2, \sigma)) = \perp$ **then**
  return $\perp$
3 $c_1 \| \cdots \| c_e \| c_{e+1} \| \cdots \| c_{n+1} = S^1 \| S^2$
4 $pad(S^1 \| Y_1) = y_1 \| \cdots \| y_j$
5 **for** $i = 1, \ldots, j$ **do**
  $(x \| w) = \pi(x \oplus y_i \| w)$
6 $K = \lfloor x \rfloor_k \oplus K_h$; $T = T_k \oplus K$
7 $x = \mathrm{ID}_S^* \| \mathrm{ID}_R \| K$; $w = \mathrm{IV}_2$
8 **for** $i = 1, \ldots, n+1$ **do**
  $(x \| w) = \pi(x \| w)$; $m_i = x \oplus c_i$; $x = c_i$
9 $(x \| w) = \pi(x \| w)$; $T' = \lfloor x \rfloor_k$
10 $X' = m_1 \| \cdots \| m_{n+1}$
11 **if** $T = T'$ **then**
  if there exists M such that $M = \mathsf{checkout}(X', r, k, \ell_{\mathrm{sg}})$ **then**
    return M
  else
    return $\perp$
  else
  return $\perp$

On π-query m, where $m \in \{0,1\}^b$

1 let $(x \| w) = m$, where $x \in \{0,1\}^r$, $w \in \{0,1\}^c$
2 **if** $(m, v) \in I_\pi$ **then**
  return v
3 $v \xleftarrow{\$} \{0,1\}^b$
4 **if** there exists m' such that $(m', v) \in I_\pi$ **then**
  $v \xleftarrow{\$} \{0,1\}^b \setminus \{v : (*, v) \in I_\pi\}$,
  where $* \in \{0,1\}^b$
5 $I_\pi = I_\pi \cup \{(m, v)\}$
6 **return** v

On π^{-1}-query v, where $v \in \{0,1\}^b$

1 **if** $(m, v) \in I_\pi$ **then**
  return m
2 $m \xleftarrow{\$} \{0,1\}^b$
3 **if** there exists v' such that $(m, v') \in I_\pi$ **then**
  $m \xleftarrow{\$} \{0,1\}^b \setminus \{m : (m, *) \in I_\pi\}$,
  where $* \in \{0,1\}^b$
4 $I_\pi = I_\pi \cup \{(m, v)\}$
5 **return** m

On $\pi_{\mathcal{A}}$-query m, where $m \in \{0,1\}^b$

1 $v = \pi(m)$
2 $I_\pi^{\mathcal{A}} = I_\pi^{\mathcal{A}} \cup \{(m, v)\}$
3 **return** v

On $\pi_{\mathcal{A}}^{-1}$-query v, where $v \in \{0,1\}^b$

1 $m = \pi^{-1}(v)$
2 $I_\pi^{\mathcal{A}} = I_\pi^{\mathcal{A}} \cup \{(m, v)\}$
3 **return** m

Figure 4: Game G0

Games G3 and G4. G3 remains the same as G2. In G3, we split up the output v of π in input rate v_1 and capacity rate v_2. We also have a set $\mathcal{L}_c$, which initially consists of IV_2 and IV_3. The output v of π is chosen at random from the previous outputs. We mark an event as $bad \leftarrow true$ in case v_2 is part of any previous output, $v_2 \in \mathcal{L}_c$. In G4, π is converted back to a permutation from a random function. Now, in G4, if $bad \leftarrow true$ happens, then v_2 is chosen again randomly from its set, but rejecting the values already in the set $\mathcal{L}_c$.

Games G1 and G2.
Initialization: $I_\pi = I_\pi^{\mathcal{A}} = \emptyset$; $\mathrm{IV}_1 = 0^r$; $\mathrm{IV}_2 = 0^c$; $\mathrm{IV}_3 = \mathrm{IV}_2 \oplus 1$; $(\mathrm{sk}_R, \mathrm{pk}_R) \leftarrow \mathsf{GenEnc}(1^k)$, $\mathrm{pk}_S^*, \mathrm{ID}_S^*$
Signlist: $\{(S_i, \sigma_i) : \sigma_i = \mathsf{Sign}_{\mathrm{sk}_S^*}(S_i)$ for all $1 \le i \le q_H$ and each S_i chosen randomly$\}$.

On π-query m, where $m \in \{0,1\}^b$

1 let $(x \parallel w) = m$, where $x \in \{0,1\}^r$, $w \in \{0,1\}^c$
2 **if** $(m, v) \in I_\pi$ **then**
  return v
3 $v \xleftarrow{\$} \{0,1\}^b$
4 **if** there exists m' such that $(m', v) \in I_\pi$ **then**
  bad $\leftarrow$ *true*; $v \xleftarrow{\$} \{0,1\}^b \setminus \{v : (*, v) \in I_\pi\}$,
  where $* \in \{0,1\}^b$
5 $I_\pi = I_\pi \cup \{(m, v)\}$
6 **return** v

On π^{-1}-query v, where $v \in \{0,1\}^b$

1 let $(v_1 \parallel v_2) = m$, where $v_1 \in \{0,1\}^r$, $v_2 \in \{0,1\}^c$
2 **if** $(m, v) \in I_\pi$ **then**
  return m
3 $m \xleftarrow{\$} \{0,1\}^b$
4 **if** there exists v' such that $(m, v') \in I_\pi$ **then**
  bad $\leftarrow$ *true*; $m \xleftarrow{\$} \{0,1\}^b \setminus \{m : (m, *) \in I_\pi\}$,
  where $* \in \{0,1\}^b$
5 $I_\pi = I_\pi \cup \{(m, v)\}$
6 **return** m

The remaining oracles are the same as G0.

Figure 5: Games G1 and G2. The dashed box has a dummy line of code with respect to G0, added and shared with both G1 and G2. G1 is with the solid box and G2 without the solid box.

Games G3 and G4.
Initialization: $I_\pi = I_\pi^{\mathcal{A}} = \emptyset$; $\mathrm{IV}_1 = 0^r$; $\mathrm{IV}_2 = 0^c$; $\mathrm{IV}_3 = \mathrm{IV}_2 \oplus 1$; $\mathcal{L}_c = \{\mathrm{IV}_2, \mathrm{IV}_3\}$;
$(\mathrm{sk}_R, \mathrm{pk}_R) \leftarrow \mathsf{GenEnc}(1^k)$, $\mathrm{pk}_S^*, \mathrm{ID}_S^*$
Signlist: $\{(S_i, \sigma_i) : \sigma_i = \mathsf{Sign}_{\mathrm{sk}_S}(S_i)$ for all $1 \le i \le q_H$ and each S_i chosen randomly$\}$.

On π-query m, where $m \in \{0,1\}^b$

1 let $(x \parallel w) = m$, where $x \in \{0,1\}^r$, $w \in \{0,1\}^c$
2 **if** $(m, v) \in I_\pi$ **then**
  return v
3 $v_1 \parallel v_2 \xleftarrow{\$} \{0,1\}^b$, where $v_1 \in \{0,1\}^r$, $v_2 \in \{0,1\}^c$,
4 **if** $v_2 \in \mathcal{L}_c \cup \{w\}$ **then**
  bad $\leftarrow$ *true*; $v_2 \xleftarrow{\$} \{0,1\}^c \setminus \mathcal{L}_c \cup \{w\}$
5 $I_\pi = I_\pi \cup \{(m, v_1 \parallel v_2)\}$; $\mathcal{L}_c = \mathcal{L}_c \cup \{v_2, w\}$
6 **return** $v = v_1 \parallel v_2$

On π^{-1}-query v, where $v \in \{0,1\}^b$

1 let $(v_1 \parallel v_2) = v$, where $v_1 \in \{0,1\}^r$, $v_2 \in \{0,1\}^c$
2 **if** $(m, v) \in I_\pi$ **then**
  return m
3 $m' \parallel m'' \xleftarrow{\$} \{0,1\}^b$, where $m' \in \{0,1\}^r$,
  $m'' \in \{0,1\}^c$,
4 **if** $m'' \in \mathcal{L}_c \cup \{v_2\}$ **then**
  bad $\leftarrow$ *true*; $m'' \xleftarrow{\$} \{0,1\}^c \setminus \mathcal{L}_c \cup \{v_2\}$
5 $I_\pi = I_\pi \cup \{(m' \parallel m'', v)\}$; $\mathcal{L}_c = \mathcal{L}_c \cup \{m'', v_2\}$
6 **return** $m = m' \parallel m''$

The remaining oracles are the same as G0.

Figure 6: Games G3 and G4. The dashed box shows a dummy line of code added in G3 with respect to G2. G3 is without the solid box and G4 with the solid box.

So, in case of *bad* $\leftarrow$ *true*, the input rate part of π's output is random and the capacity part is different from all previous capacity parts of the outputs. In G4, π works again as an ideal permutation, but the permutation is happening over the capacity parts of the output. After every query, the sets I_π and $\mathcal{L}_c$ are updated according to the input/output response of π. The probability of *bad* $\leftarrow$ *true* will be $\frac{q_\pi(q_\pi+1)}{2^c}$. Therefore, $|\Pr[G_4] - \Pr[G_3]| \le \frac{q_\pi(q_\pi+1)}{2^c}$.

From G5 to G9, we start making changes in the SignEnc oracle. We try to make the output of the SignEnc oracle random by using a random output of π. We use the message/signature pair list Signlist having q_H elements, where messages are chosen at random and signatures are calculated based on sk_{S^*}. In the last SignEnc, it can respond with random output using a pre-computed Signlist, likewise independent of $\mathsf{Sign}_{\mathrm{sk}_{S^*}}$.

Games G5 and G6. Initialize as in G4.

On SignEnc-query M, ID_R

1 $K \xleftarrow{\$} \{0,1\}^k$; $x = \mathrm{IV}_1 = \mathrm{ID}_S^* \parallel \mathrm{ID}_R \parallel 0^k$; $w = \mathrm{IV}_2$
2 $\mathsf{checkin}(M, r, k, \ell_{\mathrm{sg}}) = m_1 \parallel \cdots \parallel m_{(n+1)}$
3 $x = \mathrm{ID}_S^* \parallel \mathrm{ID}_R \parallel K$
4 $C^* \parallel T^* \xleftarrow{\$} \{0,1\}^{((n+1)*r)+r}$
5 $c_1^* \parallel c_2^* \parallel \cdots \parallel c_{n+1}^* = C^*$; $T^* = \lfloor T^* \rfloor_k$
6 **for** $i = 1 \rightarrow (n+1)$ **do**
  $(x \parallel w) = \pi(x \parallel w)$; $x = x \oplus m_i$; $c_i = x$
7 $(x \parallel w) = \pi(x \parallel w)$; $T = \lfloor x \rfloor_k$
8 $S^1 \parallel S^2 \parallel T = c_1 \parallel \cdots \parallel c_e \parallel c_{e+1} \parallel \cdots \parallel c_{n+1} \parallel T$
9 $Y_1 = \mathsf{Enc}_{\mathrm{pk}_R}(S^1)$; $\sigma = \mathsf{Sign}_{\mathrm{sk}_S^*}(S^2)$
10 $pad(S^1 \parallel Y_1) = y_1 \parallel \cdots \parallel y_j$; $x = \mathrm{IV}_1$; $w = \mathrm{IV}_3$
11 **for** $i = 1, \ldots, j$ **do**
  $(x \parallel w) = \pi(x \oplus y_i \parallel w)$
12 $K_h = \lfloor x \rfloor_k \oplus K$; $T_k = T \oplus K$
13 **return** $(K_h, Y_1, Y_2 = (S^2, \sigma), T_k)$

On π-query m, where $m \in \{0,1\}^b$

1 let $(x \parallel w) = m$, where $x \in \{0,1\}^r$, $w \in \{0,1\}^c$,
2 **if** $(m, v) \in I_\pi$ **then**
  return v
3 $v_1 \parallel v_2 \xleftarrow{\$} \{0,1\}^b$, where $v_1 \in \{0,1\}^r$, $v_2 \in \{0,1\}^c$
4 **if** $v_2 \in \mathcal{L}_c \cup \{w\}$ **then**
  $v_2 \xleftarrow{\$} \{0,1\}^c \setminus \mathcal{L}_c \cup \{w\}$
5 $I_\pi = I_\pi \cup \{(m, v_1 \parallel v_2)\}$; $\mathcal{L}_c = \mathcal{L}_c \cup \{v_2, w\}$
6 **return** $v = v_1 \parallel v_2$

On π^{-1}-query v, where $v \in \{0,1\}^b$

1 let $(v_1 \parallel v_2) = v$, where $v_1 \in \{0,1\}^r$, $v_2 \in \{0,1\}^c$
2 **if** $(m, v) \in I_\pi$ **then**
  return m
3 $m' \parallel m'' \xleftarrow{\$} \{0,1\}^b$, where $m' \in \{0,1\}^r$,
  $m'' \in \{0,1\}^c$
4 **if** $m'' \in \mathcal{L}_c \cup \{v_2\}$ **then**
  $m'' \xleftarrow{\$} \{0,1\}^c \setminus \mathcal{L}_c \cup \{v_2\}$
5 $I_\pi = I_\pi \cup \{(m' \parallel m'', v)\}$; $\mathcal{L}_c = \mathcal{L}_c \cup \{m'', v_2\}$
6 **return** $m = m' \parallel m''$

The remaining oracles are the same as G0.

Figure 7: Games G5 and G6. The dashed box shows added dummy lines of code in G6 compared to G5. G5 has the same code as G4.

Games G5 and G6. G5 is the same as G6. In G6, in SignEnc, we add a dummy random string $C^* \parallel T^*$ equivalent to the length of $C \parallel T$, shown as dashed box. G5 and G6 are the same except for some dummy lines that are added in G6 at step 4 and 5 in SignEnc. In these dummy lines, $C^* \parallel T^*$ is chosen at random, and C^* is split into c_i^*, where $1 \le i \le n+1$ and each $|c_i^*| = r$.

Games G6 and G7. In G7, we change the response of π according to c_i^* such that *SpPad.Enc* outputs $C^* \parallel T^*$ for M on K. As we already know, the r-bit part of the b-bit output of π is random. Therefore, we can replace the random output x of π with another random value $m_i \oplus c_i$. Such a change will produce $C^* \parallel T^*$ as the output response of π from its "for" loop. Now $S^1 \parallel S^2 \parallel T = C^* \parallel T^*$, and this is used for calculating encryption and signature for the final output. Here $C^* = c_1 \parallel \cdots \parallel c_e \parallel c_{e+1} \parallel \cdots \parallel c_{n+1}$, $S^1 = c_1 \parallel \cdots \parallel c_e$ and $S^2 = c_{e+1} \parallel \cdots \parallel c_{n+1}$. We store the input/output response of π, called in SignEnc, in a set I_π^{sc}.

This change of response might fail if the response of the first π call using K in SignEnc is already defined by the $\mathcal{A}$ query in I_π^A using $\pi^{\mathcal{A}}$ and publicly known IV_2 and IDs. Because if the first response of π using K in SignEnc goes collision free, then all successive responses will be new due to the permutation property. Therefore, the probability of failure of this response change in G7 for q_{sc} queries is $\frac{q_\pi^{\mathcal{A}}}{2^k}$. Therefore, $|\Pr[G_7] - \Pr[G_6]| \le q_{\mathrm{sc}} \cdot \frac{q_\pi^{\mathcal{A}}}{2^k}$.

Games G7 and G8. In G8, we chose a new message/signature pair from `Signlist` at random. We replace the chosen message S_i from `Signlist` with S^2 of π loop's (*SpPad*) output. In G8, before starting to calculate *SpPad* and after generating $C^* \parallel T^*$, we set $S^1 \parallel S^2 \parallel T = C^* \parallel T^*$. Then we replace S^2 with S_i of the (message/signature) pair list, and then again set $C^* \parallel T^* = S^1 \parallel S^2 \parallel T$. The rest of the code remains the same as G7. Here we replace a random S^2 with a random S_i of `Signlist` and calculate the rest as in G7. Because both S_i and S^2 are random, no difference will arise in games G7 and G8.

Games G6 and G7. Initialize as in G4.

On SignEnc-query M, ID_R

1 $K \xleftarrow{\$} \{0,1\}^k$; $x = \mathrm{IV}_1 = 0^k$; $w = \mathrm{IV}_2$
2 $\mathsf{checkin}(M, r, k, \ell_{\mathrm{sg}}) = m_1 \| \cdots \| m_{(n+1)}$
3 $x = \mathrm{ID}_S^* \| \mathrm{ID}_R \| K$
4 $C^* \| T_x^* \xleftarrow{\$} \{0,1\}^{((n+1)*r)+r}$
5 $c_1^* \| c_2^* \| \cdots \| c_{n+1}^* = C^*$; $T^* = \lfloor T_x^* \rfloor_k$
6 **for** $i = 1 \rightarrow (n+1)$ **do**
  $v = x \| w$;
  $(x \| w) = \pi(x \| w)$; ((($x = c_i^* \oplus m_i$) $\| w$) = $\pi(x \| w)$;
  $v' = x \| w$; $I_\pi^{\mathrm{sc}} = I_\pi^{\mathrm{sc}} \cup \{v, v'\}$; $x = x \oplus m_i$; $c_i = x$
7 $(x \| w) = \pi(x \| w)$; $T = \lfloor x \rfloor_k$ (($(T_x^*) \| w) = \pi(x \| w)$; $T^* = \lfloor T_x^* \rfloor_k$)
8 $S^1 \| S^2 \| T = c_1 \| \cdots \| c_e \| c_{e+1} \| \cdots \| c_{n+1} \| T$
9 ($S^1 \| S^2 \| T = c_1^* \| \cdots \| c_e^* \| c_{e+1}^* \| \cdots \| c_{n+1}^* \| T^*$)
10 $Y_1 = \mathsf{Enc}_{\mathrm{pk}_R}(S^1)$; $\sigma = \mathsf{Sign}_{\mathrm{sk}_S^*}(S^2)$
11 $pad(S^1 \| Y_1) = y_1 \| \cdots \| y_j$; $x = \mathrm{IV}_1$; $w = \mathrm{IV}_3$
12 **for** $i = 1, \ldots, j$ **do**
  $(x \| w) = \pi(x \oplus y_i \| w)$
13 $K_h = \lfloor x \rfloor_k \oplus K$; $T_k = T \oplus K$
14 **return** $(K_h, Y_1, Y_2 = (S^2, \sigma), T_k)$

The remaining oracles are the same as G5.

Figure 8: Games G6 and G7. G6 follows the code without the rounded box. G7 follows the code without the solid box.

Games G8 and G9. In G9, the code remains the same as in G8; instead of calculating $\sigma = \mathsf{Sign}_{\mathrm{sk}_S}(S^2 = S_i)$, one can simply replace this operation with a pre-calculated σ_i for S_i from `Signlist`. Now SignEnc is independent of sk_S of Sign and later available to adversary $\mathcal{B}$ for a uUF-RMA attack on Sign.

From now on, we start making changes in the VerDec oracle.

Game G10. In G10, we add some dummy lines, which does not affect the UF-CMA experiment of the game, and G10 remains the same as G9. In G10, we modify the VerDec oracle such that we detect an existential forgery on VerDec and show a reduction to universal forgery on Ver. Whenever we discuss a forgery, we consider $\mathrm{ID}_R = \mathrm{ID}_R^*$ in VerDec and related pk_{R^*} and sk_{R^*} for the target sender ID_{S^*}.

We set *flag* to a boolean value *old* initially, and set it to *new* in case the input/output response of π during VerDec does not belong to (I_{sc}^π and $I_\pi^{\mathcal{A}}$). Here *flag* = *old* signifies that the input to VerDec is the output of SignEnc for some i-th query in case $I_{\mathrm{vd}} \subset I_{\mathrm{sc}}^\pi$, or all of π's input/output responses are already known to adversary $\mathcal{A}$ in $I_\pi^{\mathcal{A}}$ if $I_{\mathrm{vd}} \subset I_\pi^{\mathcal{A}}$. Similarly, if *flag* becomes *new*, then one of the values of π in VerDec is new with regard to SignEnc. In case validation passed for *flag* = *new*, then essentially the answer M is not queried before SignEnc, and one of the values from K_h, Y_1, Y_2, T is used differently compared to any value in the output of SignEnc.

A forgery is assumed to be valid only when $\mathsf{Ver}_{\mathrm{pk}_S}(y_2 = (S^2, \sigma)) \neq \perp$ and $T = T'$ happens under *flag* = *new* for ID_R^*. We try to detect a forgery based on a randomly chosen known input of Ver.

Game G11. In G11, we return $\perp$ in case *flag* = *new*. Here the difference between G10 and G11 is the probability of $T = T'$ in case *flag* = *new*.

Games G8 and G9. Initialize as in G4.

On SignEnc-query M, ID_R

$K \xleftarrow{\$} \{0,1\}^k$; $w = \mathrm{IV}_2$
$\mathsf{checkin}(M, r, k, \ell_{\mathrm{sg}}) = m_1 \| \cdots \| m_{(n+1)}$
$C^* \| T^* \xleftarrow{\$} \{0,1\}^{((n+1)*r)+k}$
$c_1^* \| c_2^* \| \cdots \| c_{n+1}^* = C^*$
$S^1 \| S^2 \| T = c_1^* \| \cdots \| c_e^* \| c_{e+1}^* \| \cdots \| c_{n+1}^* \| T^*$
$i \xleftarrow{\$} \{1 \ldots q_H\}/I$; $S^2 = S_i$; $I = I \cup i$
$c_1^* \| \cdots \| c_e^* \| c_{e+1}^* \| \cdots \| c_{n+1}^* = S^1 \| S^2$
$x = \mathrm{ID}_S^* \| \mathrm{ID}_R \| K$
for $i = 1, \ldots, n+1$ **do**
  $v = x \| w$; $((x = c_i^* \oplus m_i) \| w) = \pi(x \| w)$; $v' = x \| w$; $I_\pi^{\mathrm{sc}} = I_\pi^{\mathrm{sc}} \cup \{v, v'\}$; $x = x \oplus m_i$; $c_i = x$
$(x \| w) = \pi(x \| w)$; $T = \lfloor x \rfloor_k$
$S^1 \| S^2 \| T = c_1^* \| \cdots \| c_e^* \| c_{e+1}^* \| \cdots \| c_{n+1}^* \| T^*$
$Y_1 = \mathsf{Enc}_{\mathrm{pk}_R}(S^1)$; $\sigma = \mathsf{Sign}_{\mathrm{sk}_S^*}(S^2)$; $Y_1 = \mathsf{Enc}_{\mathrm{pk}_R}(S^1)$; $\sigma = \sigma_i$
$pad(S^1 \| Y_1) = y_1 \| \cdots \| y_j$; $x = \mathrm{IV}_1$; $w = \mathrm{IV}_3$
for $i = 1, \ldots, j$ **do**
  $(x \| w) = \pi(x \oplus y_i \| w)$
$K_h = \lfloor x \rfloor_k \oplus K$; $T_k = T \oplus K$
return $(K_h, Y_1, Y_2 = (S^2, \sigma), T_k)$

The remaining oracles are the same as G5.

Figure 9: Games G8 and G9. The dashed box shows dummy lines of codes added in G8 compared to G7. G8 follows the code without the rounded box. G9 follows the code with the rounded box.

In case validation passed for *flag* = *new*, then essentially the answer M is not queried before SignEnc, and one of the values from π is freshly defined. This leads to a target collision on the proposed T in the input to VerDec. This happens with a probability of $\frac{1}{2^k}$. Therefore, $|\Pr[G_{11}] - \Pr[G_{10}]| \leq \frac{q_{\mathrm{usc}}}{2^k}$.

Game G12. G12 is the same as G11 except for some dummy lines of code added, shown in dashed boxes.

- Initially, a random S_j of length ℓ_{sg} is chosen. In case this S_j appears in SignEnc during answering a query, we abort SignEnc from answering. The probability of such happening is $\frac{q_H}{2^{\ell_{\mathrm{sg}}}}$, and this event is not helpful in the forgery because such a query does not provide any information to adversary $\mathcal{A}$.
- We also mark a dummy event bad_{sign} as *true*if, during the VerDec query, $S^2 = S_j$ and $\mathsf{Ver}_{\mathrm{pk}_{S^*}}(y_2) = \top$ for $\mathrm{ID}_R = \mathrm{ID}_R^*$. This event signifies that the adversary has provided a valid signature on a randomly chosen S^2 for a targeted ID of sender and receiver. Later, we show that the probability of such bad_{sign} being *true*is equivalent to $\mathrm{Adv}^{\text{uUF-RMA}}_{\text{SIGN},\mathcal{B}}(k)$.
- We also mark an event as $bad \leftarrow true$ in case VerDec returns M if $I_{\mathrm{vd}} \subseteq I_\pi^{\mathcal{A}}$ is true and *flag* is still *old*.

Game G13. In G13, we return $\perp$ instead of M in case $bad \leftarrow true$. We check the probability of this *bad* event happening. This event is possible in either of two cases. We denote the first case as bad_π and the second case as bad_{sign}, e.g., $\Pr[bad \leftarrow true] = \Pr[bad_\pi \leftarrow true] + \Pr[bad_{\mathrm{sign}} \leftarrow true]$.

The probability of $bad_\pi \leftarrow true$ is as follows. This is the case when the adversary has generated a valid ciphertext using an individual query to $\pi^{\mathcal{A}}$ with the help of a known message/signature pair. Adversary $\mathcal{A}$ could use custom K, pk_R^* and values of m_i so that adaptive calls of $\pi^{\mathcal{A}}$ will produce the desired S^2 with known σ and some random T. Here comes the part of special addition of an 0^r string block in message padding during checkin and checkout. This block forces $\mathcal{A}$ to select a particular K and values of m_i such that, after producing S^1, the next output of $\pi^{\mathcal{A}}$ should be equivalent to the first r-bit of S^2. This is essential to pass

Games G10 and G11. Initialize as in G4.

On SignEnc-query M, ID_R

$K \xleftarrow{\$} \{0,1\}^k$; $w = \mathrm{IV}_2$
$\mathsf{checkin}(M, r, k, \ell_{\mathrm{sg}}) = m_1 \| \cdots \| m_{(n+1)}$
$C^* \| T^* \xleftarrow{\$} \{0,1\}^{((n+1)*r)+k}$
$c_1^* \| c_2^* \| \cdots \| c_{n+1}^* = C^*$
$S^1 \| S^2 \| T = c_1^* \| \cdots \| c_e^* \| c_{e+1}^* \| \cdots \| c_{n+1}^* \| T^*$
$i \xleftarrow{\$} \{1 \ldots q_H\}/I$; $S^2 = S_i$; $I = I \cup i$
$c_1^* \| \cdots \| c_e^* \| c_{e+1}^* \| \cdots \| c_{n+1}^* = S^1 \| S^2$
$x = \mathrm{ID}_S^* \| \mathrm{ID}_R \| K$
for $i = 1, \ldots, n+1$ **do**
  $v = x \| w$; $((x = c_i^* \oplus m_i) \| w) = \pi(x \| w)$;
  $v' = x \| w$; $I_\pi^{\mathrm{sc}} = I_\pi^{\mathrm{sc}} \cup \{v, v'\}$; $x = x \oplus m_i$; $c_i = x$
$(x \| w) = \pi(x \| w)$; $T = \lfloor x \rfloor_k$
$S^1 \| S^2 \| T = c_1^* \| \cdots \| c_e^* \| c_{e+1}^* \| \cdots \| c_{n+1}^* \| T^*$
$Y_1 = \mathsf{Enc}_{\mathrm{pk}_R}(S^1)$; $\sigma = \sigma_i$
$pad(S^1 \| Y_1) = y_1 \| \cdots \| y_j$; $x = \mathrm{IV}_1$; $w = \mathrm{IV}_3$
for $i = 1, \ldots, j$ **do**
  $(x \| w) = \pi(x \oplus y_i \| w)$
$K_h = \lfloor x \rfloor_k \oplus K$; $T_k = T \oplus K$
return $(K_h, Y_1, Y_2 = (S^2, \sigma), T_k)$

On VerDec-query (K_h, Y_1, Y_2, T_k)

$S^1 = \mathsf{Dec}_{\mathrm{sk}_R}(Y_1)$; $x = \mathrm{IV}_1$; $w = \mathrm{IV}_3$
if $\mathsf{Ver}_{\mathrm{pk}_{S^*}}(Y_2 = (S^2, \sigma)) = \perp$ **then**
  return $\perp$
$c_1 \| \cdots \| c_e \| c_{e+1} \| \cdots \| c_{n+1} = S^1 \| S^2$
$pad(S^1 \| Y_1) = y_1 \| \cdots \| y_j$; $I_{\mathrm{vd}} = \emptyset$
for $i = 1, \ldots, j$ **do**
  $v = x \| w$; $(x \| w) = \pi(x \| w)$; $v' = x \| w$;
  $I_{\mathrm{vd}} = \{v, v'\} \cup I_{\mathrm{vd}}$
$K = \lfloor x \rfloor_k \oplus K_h$; $T = T_k \oplus K$
$x = \mathrm{ID}_S^* \| \mathrm{ID}_R \| K$; $w = \mathrm{IV}_2$; $flag \leftarrow old$
for $i = 1 \rightarrow (n+1)$ **do**
  $v = x \| w$; $(x \| w) = \pi(x \| w)$; $v' = x \| w$;
  $m_i = x \oplus c_i$; $x = c_i$; $I_{\mathrm{vd}} = \{v, v'\} \cup I_{\mathrm{vd}}$
$v = x \| w$; $(x \| w) = \pi(x \| w)$
$v' = x \| w$; $T' = \lfloor x \rfloor_r$; $I_{\mathrm{vd}} = \{v, v'\} \cup I_{\mathrm{vd}}$
$X' = m_1 \| \cdots \| m_{n+1}$
if $I_{\mathrm{vd}} \not\subset I_{\mathrm{sc}}$ and $I_{\mathrm{vd}} \not\subset I_\pi^{\mathcal{A}}$ **then**
  $flag \leftarrow new$
if $T = T'$ and $flag = new$ **then**
  if there exists M such that
  $M = \mathsf{checkout}(X', r, k, \ell_{\mathrm{sg}})$ **then**
    return M **return** $\perp$
  else
    return $\perp$
else
  return $\perp$
if $T = T'$ and $flag = old$ **then**
  if there exists M such that
  $M = \mathsf{checkout}(X', r, k, \ell_{\mathrm{sg}})$ **then**
    return M
  else
    return $\perp$
else
  return $\perp$

The remaining oracles are the same as G5.

Figure 10: Games G10 and G11. The dashed boxes show dummy lines of code added in G10 compared to G9. G10 follows the code with the rounded box, without the solid box. G11 follows the code without the rounded box, with the solid box.

Games G12 and G13. Initialize as in G4 and choose an $S_j \xleftarrow{\$} \{0,1\}^{\ell_{sg}}$.

On SignEnc-query M, ID_R

$K \xleftarrow{\$} \{0,1\}^k$; $w = \mathrm{IV}_2$
checkin$(M, r, k, \ell_{sg}) = m_1 \| \cdots \| m_{(n+1)}$
$C^* \| T^* \xleftarrow{\$} \{0,1\}^{((n+1)*r)+k}$
$c_1^* \| c_2^* \| \cdots \| c_{n+1}^* = C^*$
$S^1 \| S^2 \| T = c_1^* \| \cdots \| c_e^* \| c_{e+1}^* \| \cdots \| c_{n+1}^* \| T^*$
$i \xleftarrow{\$} \{1 \ldots q_H\}/I; S^2 = S_i; I = I \cup i$
$c_1^* \| \cdots \| c_e^* \| c_{e+1}^* \| \cdots \| c_{n+1}^* = S^1 \| S^2$
$x = \mathrm{ID}_S^* \| \mathrm{ID}_R \| K$
for $i = 1, \ldots, n+1$ **do**
  $v = x \| w$; $((x = c_i^* \oplus m_i) \| w) = \pi(x \| w)$;
  $v' = x \| w$; $I_\pi^{sc} = I_\pi^{sc} \cup \{v, v'\}$; $x = x \oplus m_i$; $c_i = x$
$(x \| w) = \pi(x \| w)$; $T = \lfloor x \rfloor_k$
$S^1 \| S^2 \| T = c_1^* \| \cdots \| c_e^* \| c_{e+1}^* \| \cdots \| c_{n+1}^* \| T^*$
if $S_i = S_j$ **then** abort
$Y_1 = \mathsf{Enc}_{\mathrm{pk}_R}(S^1)$; $\sigma = \sigma_i$
$pad(S^1 \| Y_1) = y_1 \| \cdots \| y_j$; $x = \mathrm{IV}_1$; $w = \mathrm{IV}_3$
for $i = 1, \ldots, j$ **do**
  $(x \| w) = \pi(x \oplus y_i \| w)$
$K_h = \lfloor x \rfloor_k \oplus K$; $T_k = T \oplus K$
return $(K_h, Y_1, Y_2 = (S^2, \sigma), T_k)$

On VerDec-query (K_h, Y_1, Y_2, T_k)

$S^1 = \mathsf{Dec}_{\mathrm{sk}_R}(Y_1)$; $x = \mathrm{IV}_1$; $w = \mathrm{IV}_3$
if $\mathsf{Ver}_{\mathrm{pk}_{S^*}}(Y_2 = (S^2, \sigma)) = \perp$ **then**
  return $\perp$
if $S^2 = S_j$ *and* $\mathrm{ID}_R = \mathrm{ID}_R^*$ **then**
  $bad_{\mathrm{sign}} \leftarrow true$
$c_1 \| \cdots \| c_e \| c_{e+1} \| \cdots \| c_{n+1} = S^1 \| S^2$
$pad(S^1 \| Y_1) = y_1 \| \cdots \| y_j$; $I_{\mathrm{vd}} = \emptyset$; $flag \leftarrow old$
for $i = 1, \ldots, j$ **do**
  $(x \| w) = \pi(x \oplus y_i \| w)$
$K = \lfloor x \rfloor_k \oplus K_h$; $T = T_k \oplus K$
$x = \mathrm{ID}_S^* \| \mathrm{ID}_R \| K$; $w = \mathrm{IV}_2$
for $i = 1 \to (n+1)$ **do**
  $v = x \| w$; $(x \| w) = \pi(x \| w)$; $v' = x \| w$;
  $m_i = x \oplus c_i$; $x = c_i$; $I_{\mathrm{vd}} = \{v, v'\} \cup I_{\mathrm{vd}}$
$v = x \| w$; $(x \| w) = \pi(x \| w)$
$v' = x \| w$; $T' = \lfloor x \rfloor_r$; $I_{\mathrm{vd}} = \{v, v'\} \cup I_{\mathrm{vd}}$
$X' = m_1 \| \cdots \| m_{n+1}$
if $I_{\mathrm{vd}} \notin I_{\mathrm{sc}}$ and $I_{\mathrm{vd}} \notin I_\pi^{\mathcal{A}}$ **then**
  $flag \leftarrow new$
if $T = T'$ and $flag = old$ **then**
  if $I_{\mathrm{vd}} \subseteq I_\pi^{\mathcal{A}}$ **then**
    if there exists M such that
    $M = \mathsf{checkout}(X', r, k, \ell_{sg})$ **then**
      $bad \leftarrow true$;
      return M $\perp$
    else
      return $\perp$
  if there exist M such that
  $M = \mathsf{checkout}(X', r, k, \ell_{sg})$ **then**
    return M
  else
    return $\perp$
else
  return $\perp$

The remaining oracles are the same as G5.

Figure 11: Games G12 and G13. The dashed boxes show added dummy lines of code in G12 compared to G11. G12 follows the code without the solid box, with the rounded box. G13 follows the code with the solid box, without the rounded box.

the checkout function. The probability of this happening is $\frac{q_{sc}}{2^r}$ for the available q_{sc} message/signature pairs through SignEnc queries.

The probability of $bad_{\mathrm{sign}} \leftarrow true$ is as follows. This case happens when the adversary has generated a valid ciphertext using $I_\pi^{\mathcal{A}}$ having known an individual query to $\pi^{\mathcal{A}}$ and *without* the help of a known message/signature pair by generating a valid signature for random S^2. This could happen as follows: The adversary $\mathcal{A}$ asks queries to π for some random K, pk_R^* and custom m_i's according to the checkout function to generate

Game G14.

Initialization: $I_\pi^{\text{sc}} = I = I_\pi = I_\pi^{\mathcal{A}} = \emptyset$; $\text{IV}_1 = 0^r$; $\text{IV}_2 = 0^c$; $(\text{sk}_R, \text{pk}_R) \leftarrow \text{GenEnc}(1^k)$, $\text{pk}_S^*, \text{ID}_S^*$
Signlist: $\{(S_i, \sigma_i) : \sigma_i = \text{Sign}_{\text{sk}_S}(S_i)$ for all $1 \le i \le q_H$ and each S_i chosen randomly$\}$ and choose an $S_j \xleftarrow{\$} \{0,1\}^{\ell_{\text{sg}}}$

On SignEnc-query M, ID_R

$K \xleftarrow{\$} \{0,1\}^k$; $w = \text{IV}_2$
checkin$(M, r, k, \ell_{\text{sg}}) = m_1 \| \cdots \| m_{(n+1)}$
$C^* \| T^* \xleftarrow{\$} \{0,1\}^{((n+1)*r)+k}$
$c_1^* \| c_2^* \| \cdots \| c_{n+1}^* = C^*$
$S^1 \| S^2 \| T = c_1^* \| \cdots \| c_e^* \| c_{e+1}^* \| \cdots \| c_{n+1}^* \| T^*$
$i \xleftarrow{\$} \{1 \ldots q_H\}/I$; $S^2 = S_i$; $I = I \cup i$
$c_1^* \| \cdots \| c_e^* \| c_{e+1}^* \| \cdots \| c_{n+1}^* = S^1 \| S^2$
$x = \text{ID}_S^* \| \text{ID}_R \| K$
for $i = 1, \ldots, n+1$ **do**
  $v = x \| w$; $((x = c_i^* \oplus m_i) \| w) = \pi(x \| w)$;
  $v' = x \| w$; $I_\pi^{\text{sc}} = I_\pi^{\text{sc}} \cup \{v, v'\}$; $x = x \oplus m_i$; $c_i = x$
$(x \| w) = \pi(x \| w)$; $T = \lfloor x \rfloor_k$
$S^1 \| S^2 \| T = c_1^* \| \cdots \| c_e^* \| c_{e+1}^* \| \cdots \| c_{n+1}^* \| T^*$
if $S_i = S_j$ **then** abort
$Y_1 = \text{Enc}_{\text{pk}_R}(S^1)$; $\sigma = \sigma_i$;
$pad(S^1 \| Y_1) = y_1 \| \cdots \| y_j$; $x = \text{IV}_1$; $w = \text{IV}_3$
for $i = 1, \ldots, j$ **do**
  $(x \| w) = \pi(x \oplus y_i \| w)$
$K_h = \lfloor x \rfloor_k \oplus K$; $T_k = T \oplus K$
return $(K_h, Y_1, Y_2 = (S^2, \sigma), T_k)$

On VerDec-query (K_h, Y_1, Y_2, T_k)

$S^1 = \text{Dec}_{\text{sk}_R}(Y_1)$; $x = \text{IV}_1$; $w = \text{IV}_3$
if $\text{Ver}_{\text{pk}_S}(Y_2 = (S^2, \sigma)) = \perp$ **then**
  return $\perp$
if $S^2 = S_j$ and $\text{ID}_R = \text{ID}_R^*$ **then**
  $bad_{\text{sign}} \leftarrow true$; $\sigma^* = \sigma$
$c_1 \| \cdots \| c_e \| c_{e+1} \| \cdots \| c_{n+1} = S^1 \| S^2$
$pad(S^1 \| Y_1) = y_1 \| \cdots \| y_j$; $I_{\text{vd}} = \emptyset$
for $i = 1, \ldots, j$ **do**
  $(x \| w) = \pi(x \oplus y_i \| w)$
$K = \lfloor x \rfloor_k \oplus K_h$; $T = T_k \oplus K$
$x = \text{ID}_S^* \| \text{ID}_R \| K$; $flag \leftarrow old$
for $i = 1 \to (n+1)$ **do**
  $v = x \| w$; $(x \| w) = \pi(x \| w)$; $v' = x \| w$;
  $m_i = x \oplus c_i$; $x = c_i$; $I_{\text{vd}} = \{v, v'\} \cup I_{\text{vd}}$
$v = x \| w$; $(x \| w) = \pi(x \| w)$
$v' = x \| w$; $T' = \lfloor x \rfloor_r$; $I_{\text{vd}} = \{v, v'\} \cup I_{\text{vd}}$
$X' = m_1 \| \cdots \| m_{n+1}$
if $(I_{\text{vd}} \not\subset I_{\text{sc}}$ and $I_{\text{vd}} \not\subset I_\pi^{\mathcal{A}})$ or $I_{\text{vd}} \subseteq I_\pi^{\mathcal{A}}$ **then**
  $flag \leftarrow new$
if $T = T'$ and $flag = old$ **then**
  if there exists M such that $M = \text{checkout}(X', r, k, \ell_{\text{sg}})$ **then**
    return M
  else
    return $\perp$
else
  return $\perp$

The remaining oracles are the same as G5.

Figure 12: Game G14. This is the same as G13 with simplified code.

random K_h, S^1, S^2 and T_k, which will also validate $T = T'$ upon verification. Now, in order to pass the validation of Sign, $\mathcal{A}$ needs to have a valid signature over Sign for random S^2. Because $\mathcal{A}$ knows the targeted message before generating the signature, this becomes equivalent to universal forgery for a random message. Therefore, $\Pr[bad_{\text{sign}}] \le \text{Adv}_{\text{SIGN}}^{\text{uUF-RMA}}(k)$.

Therefore, the adversary needs to produce either a collision over the r-bit of S^2 using a π query and known (S^2, σ) or, alternatively, produce a valid signature over random S^2, which is the output of π queries. Therefore, if the adversary passes the checkout function, then essentially $\mathcal{A}$ produces the collision. The probability of such a collision happening is $\frac{q_{\text{sc}}}{2^r} + \Pr[bad_{\text{sign}}]$. Therefore, $|\Pr[G_{13}] - \Pr[G_{12}]| \le \frac{q_{\text{sc}}}{2^r} + \text{Adv}_{\text{SIGN}}^{\text{uUF-RMA}}(k)$.

Games G14 and G13. G14 is the same as G13. G14 is the final ideal game, and we simplify the cases by merging the *bad* event with the $flag \leftarrow new$ event because, in both events, VerDec is returning $\perp$. Now *flag* is set to *new*in case $(I_{\text{vd}} \not\subset I_{\text{sc}}$ and $I_{\text{vd}} \not\subset I_\pi^{\mathcal{A}})$ or $I_{\text{vd}} \subseteq I_\pi^{\mathcal{A}}$, and VerDec returns $\perp$ if *flag* is *new*. Return of M will happen only if *flag* is *old*and validation of T passed. Now, essentially, $\mathcal{A}$ will get $\perp$ for all his queries unless

Adversary $\mathcal{B}$.
Initialization: given ID_S^* and ID_R^*, public/private key pair of R^* as pk_{R^*} and sk_{R^*}.
The public key of S^* is pk_{S^*} and the target message is M^*. We also denote M^* as μ or S_j.

Adversary $\mathcal{A}$.
Initialization: $I_\pi^{\mathrm{sc}} = I = I_\pi = I_\pi^{\mathcal{A}} = \emptyset$; $\mathrm{IV}_1 = 0^r$; $\mathrm{IV}_2 = 0^c$; $(\mathrm{sk}_R, \mathrm{pk}_R) \leftarrow \mathsf{GenEnc}(1^k)$, pk_S^*, ID_S^*
`Signlist`: $\{(S_i, \sigma_i) : \sigma_i = \mathsf{Sign}_{\mathrm{sk}_S}(S_i)$ for all $1 \le i \le q_H$ and each S_i chosen randomly$\}$

The remaining oracles are the same as G14.

Finalization: If VerDec return M and $bad_{\mathrm{sign}} \leftarrow true$ then return σ^*.

Figure 13: Adversary $\mathcal{B}$ over uUF-RMA

either he produces a valid signature on any random S^2 not queried before or he queries the output of $\mathsf{SignEnc}$. □

B Proof sketch of Lemma 4.6

Proof of Lemma 4.6. We consider the following experiment for IND-CCA security of SIGNCRYPT against adversary $\mathcal{A}$.

Experiment. $\mathrm{Exp}^{\text{IND-CCA},d}_{\text{SIGNCRYPT},\mathcal{A}}(k)$

1 $(\mathrm{sk}_{R^*}, \mathrm{pk}_{R^*}) \leftarrow \mathsf{GenEnc}_{R^*}(1^k)$
2 $(M_0, M_1, \mathrm{ID}_{S^*}) \leftarrow \mathcal{A}^{\mathsf{VerDec}_{\mathrm{sk}_{R^*}}(\cdot,\cdot),\pi(\cdot)}(\mathrm{pk}_{R^*})$
3 map to pk_{S^*} using ID_{S^*}, where $(\mathrm{sk}_{S^*}, \mathrm{pk}_{S^*}) \leftarrow \mathsf{GenSig}_S(1^k)$
4 $d \xleftarrow{\$} \{0, 1\}$
5 $y^* \leftarrow \mathsf{SignEnc}_{\mathrm{pk}_{R^*}, \mathrm{sk}_{S^*}}(M_d)$
6 $d' \leftarrow \mathcal{A}^{\mathsf{VerDec}_{\mathrm{sk}_{R^*}}(\cdot,\cdot),\pi(\cdot)}(\mathrm{pk}_{R^*}, \mathrm{sk}_{S^*}, \mathrm{pk}_{S^*}, y^*)$
7 **if** $d = d'$ and the query (y^*, ID_{S^*}) is never made to the $\mathsf{VerDec}_{\mathrm{sk}_{R^*}}(\cdot,\cdot)$ oracle **then**
  return 1
else
  return 0

The advantage of adversary $\mathcal{A}$ is given by the probability

$$\mathrm{Adv}^{\text{IND-CCA}}_{\text{SIGNCRYPT}}(k) = |\Pr[\mathrm{Exp}^{\text{IND-CCA},d}_{\text{SIGNCRYPT},\mathcal{A}}(k) = 1 \mid d \xleftarrow{\$} \{0, 1\}] - \tfrac{1}{2}|.$$

We are dealing with the insider security model in the multi-user setting; the adversary has a target receiver ID_R^* in mind. The adversary knows the receiver's public key pk_R and has access to the VerDec oracle under sk_R. Further, we assume that an adversary $\mathcal{A}$ observed q_{usc} queries to the VerDec oracle. Adversary $\mathcal{A}$ has also chosen a pair of messages M_0 and M_1 and a key pair $(\mathrm{sk}_S, \mathrm{pk}_S)$ for ID_S. It receives a ciphertext (y_1^*, y_2^*, y_3^*) under $(\mathrm{sk}_S, \mathrm{pk}_R)$ of either M_0 or M_1. The unknown message is denoted by M_d, where d is the bit the adversary wishes to find out. Adversary $\mathcal{A}$ outputs a bit d', which is equal to d with the advantage ϵ, i.e., $\Pr[d' = d] = \frac{1}{2} + \epsilon$. In the following, we use an asterisk for all internal values used in computing the challenged signcryption.

We will use game-playing techniques [9, 10]. We start from the original CCA game $\mathrm{Exp}_{\text{SIGNCRYPT},\mathcal{A}}$ or $(\mathrm{Exp}^{\text{IND-CCA},d}_{\text{SIGNCRYPT},\mathcal{A}}(k) = 1 \mid d \xleftarrow{\$} \{0, 1\})$, which denotes the event that $\mathcal{A}$ outputs $d' = d$, where $d \xleftarrow{\$} \{0, 1\}$. We want to show that

$$|\Pr[\mathrm{Exp}_{\text{SIGNCRYPT},\mathcal{A}}]| = \tfrac{1}{2} + \mathrm{negl}(k),$$

where $\mathrm{negl}(\cdot)$ is a negligible function and $\mathrm{negl}(k) \le \mathrm{Adv}^{\text{IND-CCA}}_{\text{SIGNCRYPT}}(k)$. In each game, the following set is maintained: I by π, $I_\pi^{\mathcal{A}}$ by $\pi_{\mathcal{A}}$, and Y stores the capacity c-bit values upon each query to π.

We modify SIGNCRYPT into a sequence of games G0, G1, ..., G12 such that

$$\begin{aligned}\Pr[\mathrm{Exp}_{\mathcal{F}\text{-SpAEP},\mathcal{A}}] &= \Pr[\mathrm{Exp}_{\mathrm{G0},\mathcal{A}}],\\ \Pr[\mathrm{Exp}_{\mathrm{G}(i-1),\mathcal{A}}] &= \Pr[\mathrm{Exp}_{\mathrm{G}i,\mathcal{A}}] + \mathrm{negl}(k) \quad \text{for all } 1 \le i \le 11,\\ \Pr[\mathrm{Exp}_{\mathrm{G12},\mathcal{A}}] &= \tfrac{1}{2}.\end{aligned}$$

Games G0 to G5. From G0 to G5, exactly the same changes follow as in the proof of Lemma 4.2. Therefore,

$$|\Pr[\mathrm{Exp}_{\mathrm{G0},\mathcal{A}}] - \Pr[\mathrm{Exp}_{\mathrm{G5},\mathcal{A}}]| \le \frac{(q_\pi - 1)q_\pi}{2^{b+1}} + \frac{q_\pi(q_\pi + 1)}{2^c}.$$

In G5, the game maintains an extra set I_{enc}, which stores the input/output response of π (as π_{enc}) during the SignEnc challenge query.

Games G5 and G6. Both games are the same. In G6, a dummy operation of $flag \leftarrow new$ is added in the VerDec oracle to denote a new query. The query is new in the sense that neither the query nor any part of the query during internal calls to π was queried earlier by the adversary. That is, $flag \leftarrow new$ if any π's responses are not in $I_\pi^{\mathcal{A}}$. Now the code of G6 can check the condition $T = T'$ in the cases $flag = new$ and $flag = old$ separately. If $T = T'$ and $flag = new$, then we mark this event as $bad_\pi \leftarrow true$ because bad_π is just a dummy event and the return of VerDec in G6 is not affected. Therefore, $|\Pr[\mathrm{Exp}_{\mathrm{G6},\mathcal{A}}]| = |\Pr[\mathrm{Exp}_{\mathrm{G5},\mathcal{A}}]|$.

Games G6 and G7. In G7, in VerDec, we return $\perp$ instead of M in case bad_π is true. Therefore,

$$|\Pr[\mathrm{Exp}_{\mathrm{G7},\mathcal{A}}] - \Pr[\mathrm{Exp}_{\mathrm{G6},\mathcal{A}}]| \le \Pr[bad_\pi \leftarrow true].$$

Let $(v_1 \| v_2) = \pi(x \| w)$, where $x, v_1 \in \{0,1\}^r$ and $w, v_2 \in \{0,1\}^c$. In VerDec, an input is a *new* query to π when $((x \| w), (v_1 \| v_2)) \notin I_\pi^{\mathcal{A}}$ and an *old* query when $((x \| w), (v_1 \| v_2)) \in I_\pi^{\mathcal{A}}$. If a *new* query $(x \| w)$ is input to π during VerDec, then π outputs $v_1 \| v_2$, where $v_2 \notin \mathcal{L}_c$. That is, v_2 is also new. Since v_2 is unseen so far, this ensures that the input to the next call of π is certainly new. Further, since v_2 is new, the next input $x' \| v_2$ satisfies the condition $(x' \| v_2, *) \notin I_\pi^{\mathcal{A}}$, where $*$ stands for any b-bit value. Therefore, one *new* query makes all subsequent inputs to $\pi(\cdot)$ new. We already know that, for any *new* query, the r-bit response of π is random. Therefore, in case $flag = new$, the probability of $T = T'$ is equivalent to a collision over the k-bit T value. Therefore, $\Pr[bad_\pi \leftarrow true] = \frac{q_{\text{usc}}}{2^k}$ for q_{usc} VerDec queries. Therefore,

$$|\Pr[\mathrm{Exp}_{\mathrm{G7},\mathcal{A}}] - \Pr[\mathrm{Exp}_{\mathrm{G6},\mathcal{A}}]| \le \frac{q_{\text{usc}}}{2^k}.$$

Now if this *bad* event does not happen, then G7 will return M only in case all π responses are already known to $\mathcal{A}$. Consecutively, $\mathcal{A}$ already knows the answer of VerDec with the help of π queries and available Sign, Ver and Enc functions.

Game G8. Both games are the same. G7 and G8 both return $\perp$ when a *new* query is given to the VerDec oracle. In G8, a message M is returned only when all the input/output relations of π, which would be possible during the encryption of M, are already in $I_\pi^{\mathcal{A}}$. G8 iterates over all the possible input/output pairs of $\pi \in I_\pi^{\mathcal{A}}$, initially using IV_1 and IV_3, and tries to find an S^1 such that $\mathcal{O}^{\text{PC}}(S^1, Y^1) = 1$. In the positive case, it further calculates K and then tries to find all pairs of input/output responses, which reach to T via K, S^1, S^2. If any of the responses is missing, then VerDec simply rejects the query. *Due to the insider model, a faithful assumption on the signing algorithm is that, for the same input, two different signatures cannot be generated. We will discuss the impact of this assumption later, after the proof.*

Games G8 and G9. We start incremental changes in the SIGNCRYPT oracle from G9. In G9, K^* is chosen before the signcryption query and after the "find" stage. In both cases, K^* remains random, and therefore

$$|\Pr[\mathrm{Exp}_{\mathrm{G9},\mathcal{A}}]| = \Pr[\mathrm{Exp}_{\mathrm{G8},\mathcal{A}}].$$

Some extra dummy variables S^{1*}, S^{2*}, T^*, along with K^*, are also chosen after the find stage but not used. A dummy value Y_1^* is calculated on S^{1*} using Enc.

Game G5.

Initialization: $I_{\text{enc}} = I_\pi = I_\pi^{\mathcal{A}} = \emptyset$; $\text{IV}_1 = 0^r$; $\text{IV}_2 = 0^c$; $\text{IV}_3 = \text{IV}_2 \oplus 1$; $\mathcal{L}_c = \{\text{IV}_2, \text{IV}_3\}$;
$(\text{sk}_S, \text{pk}_S) \leftarrow \text{GenEnc}(1^k)$, $\text{pk}_R^*, \text{ID}_R^*$

On SignEnc-query M_d for ID_S^*

$K^* \xleftarrow{\$} \{0,1\}^k$; $x = \text{IV}_1 = \text{ID}_S^* \parallel \text{ID}_R^* \parallel 0^k$; $w = \text{IV}_2$
checkin$(M, r, k, \ell_{\text{sg}}) = m_1 \parallel \cdots \parallel m_{(n+1)}$
$x = \text{ID}_S^* \parallel \text{ID}_R^* \parallel K$
for $i = 1 \rightarrow (n+1)$ **do**
  $(x \parallel w) = \pi_{\text{enc}}(x \parallel w)$; $x = x \oplus m_i$; $c_i^* = x$
$(x \parallel w) = \pi_{\text{enc}}(x \parallel w)$; $T^* = \lfloor x \rfloor_k$
$S^{1*} \parallel S^{2*} \parallel T^* = c_1^* \parallel \cdots \parallel c_e^* \parallel c_{e+1}^* \parallel \cdots \parallel c_{n+1}^* \parallel T^*$
$Y_1^* = \text{Enc}_{\text{pk}_R^*}(S^1)$; $\sigma^* = \text{Sign}_{\text{sk}_S^*}(S^{2*})$
$pad(S^{1*} \parallel Y_1^*) = y_1^* \parallel \cdots \parallel y_j^*$; $x = \text{IV}_1$; $w = \text{IV}_3$
for $i = 1, \ldots, j$ **do**
  $(x \parallel w) = \pi_{\text{enc}}(x \oplus y_i^* \parallel w)$
$K_h^* = \lfloor x \rfloor_k \oplus K^*$; $T_k^* = T^* \oplus K^*$
return $(K_h^*, Y_1^*, Y_2^* = (S^{2*}, \sigma^*), T_k^*)$

On VerDec-query (K_h, Y_1, Y_2, T_k)

$S^1 = \text{Dec}_{\text{sk}_{R^*}}(y_1)$; $x = \text{IV}_1$; $w = \text{IV}_3$
if $\text{Ver}_{\text{pk}_S}(Y_2 = (S^2, \sigma)) = \perp$ **then**
  return $\perp$
$c_1 \parallel \cdots \parallel c_e \parallel c_{e+1} \parallel \cdots \parallel c_{n+1} = S^1 \parallel S^2$
$y_1 \parallel \cdots \parallel y_j = pad(S^1 \parallel Y_1)$
for $i = 1, \ldots, j$ **do**
  $(x \parallel w) = \pi(x \oplus y_i \parallel w)$
$K = \lfloor x \rfloor_k \oplus K_h$; $T = T_k \oplus K$
$x = \text{ID}_S \parallel \text{ID}_R^* \parallel K$; $w = \text{IV}_2$
for $i = 1, \ldots, n+1$ **do**
  $(x \parallel w) = \pi(x \parallel w)$; $m_i = x \oplus c_i$; $x = c_i$
$(x \parallel w) = \pi(x \parallel w)$; $T' = \lfloor x \rfloor_k$
$X' = m_1 \parallel \cdots \parallel m_{n+1}$
if $T = T'$ **then**
  if there exists M such that
  $M = \text{checkout}(X', r, k, \ell_{\text{sg}})$ **then**
    return M
  else
    return $\perp$
else
  return $\perp$

On π-query m, where $m \in \{0,1\}^b$

let $(x \parallel w) = m$, where $x \in \{0,1\}^r$, $w \in \{0,1\}^c$
if $(m, v) \in I_\pi$ **then**
  return v
$v_1 \parallel v_2 \xleftarrow{\$} \{0,1\}^b$, where $v_1 \in \{0,1\}^r$, $v_2 \in \{0,1\}^c$
if $v_2 \in Y \cup \{w\}$ **then**
  $v_2 \xleftarrow{\$} \{0,1\}^c \setminus Y \cup \{w\}$
$I_\pi = I_\pi \cup \{(m, v_1 \parallel v_2)\}$; $Y = Y \cup \{v_2, w\}$
return $v = v_1 \parallel v_2$

On π^{-1}-query v, where $v \in \{0,1\}^b$

let $(v_1 \parallel v_2) = v$, where $v_1 \in \{0,1\}^r$, $v_2 \in \{0,1\}^c$
if $(m, v) \in I_\pi$ **then**
  return m
$m' \parallel m'' \xleftarrow{\$} \{0,1\}^b$, where $m' \in \{0,1\}^r$, $m'' \in \{0,1\}^c$
if $m'' \in Y \cup \{v_2\}$ **then**
  $m'' \xleftarrow{\$} \{0,1\}^c \setminus Y \cup \{v_2\}$
$I_\pi = I_\pi \cup \{(m' \parallel m'', v)\}$; $Y = Y \cup \{m'', v_2\}$
return $m = m' \parallel m''$

On π_{enc}-query m

$v = \pi(m)$
$I_{\text{enc}} = I_{\text{enc}} \cup (m, v)$

On $\pi_{\mathcal{A}}$-query m

same as G0

On $\pi_{\mathcal{A}}^{-1}$-query v

same as G0

Figure 14: Game G5

Games (G6) and [G7].

Initialization: $I_{\text{enc}} = I_\pi = I_\pi^{\mathcal{A}} = \emptyset$; $\mathsf{GenEnc}(1^k)$, $\text{pk}_R^*, \text{ID}_R^*$; $\text{IV}_1 = 0^r$; $\text{IV}_2 = 0^c$; $\text{IV}_3 = \text{IV}_2 \oplus 1$; $\mathcal{L}_c = \{\text{IV}_2, \text{IV}_3\}$; *flag* $\in \{new, old\}$

On decryption query K_h, Y_1, Y_2, T_k

1 $S^1 = \mathsf{Dec}_{\text{sk}}(Y_1)$; $x = \text{IV}_1$; $w = \text{IV}_3$
2 **if** $\mathsf{Ver}_{\text{pk}_S}(Y_2 = (S^2, \sigma)) = \perp$ **then**
  return $\perp$
3 $c_1 \| \cdots \| c_e \| c_{e+1} \| \cdots \| c_{n+1} = S^1 \| S^2$
4 $pad(S^1 \| Y_1) = y_1 \| \cdots \| y_j$; *flag* $\leftarrow$ *old*
5 **for** $i = 1, \ldots, j$ **do**
  $x = x \oplus y_i$;
  if $\{x \| w, *\} \notin I_\pi^{\mathcal{A}}$ **then** *flag* $\leftarrow$ *new*;
  $(x \| w) = \pi(x \| w)$
6 $h = \lfloor x \rfloor_k$; $K = h \oplus K_h$; $T = T_k \oplus K$
7 $x = \text{ID}_S \| \text{ID}_R \| K$; $w = \text{IV}_2$
8 **for** $i = 1 \rightarrow (n+1)$ **do**
  if $\{x \| w, *\} \notin I_\pi^{\mathcal{A}}$ **then** *flag* $\leftarrow$ *new*;
  $(x \| w) = \pi(x \| w)$; $m_i = x \oplus c_i$; $x = c_i$
9 **if** $\{x \| w, *\} \notin I_\pi^{\mathcal{A}}$ **then** *flag* $\leftarrow$ *new*
10 $(x \| w) = \pi(x \| w)$; $T' = \lfloor x \rfloor_k$; $X' = m_1 \| \cdots \| m_{n+1}$
11 **if** $T = T'$ and *flag* = *new* **then**
  $bad_\pi \leftarrow true$
  if there exists M such that $M = \mathsf{checkout}(X', r, k, \ell_{\text{sg}})$ **then**
    (**return** M) [**return** $\perp$]
  else
    return $\perp$
12 **if** $T = T'$ and *flag* = *old* **then**
  if there exists M such that $M = \mathsf{checkout}(X', r, k, \ell_{\text{sg}})$ **then**
    return M
  else
    return $\perp$
  else
    return $\perp$

The remaining oracles are the same as G5.

Figure 15: Games G6 and G7. G6 includes dummy lines, shown in dashed boxes, compared to G5 along with the rounded box. G7 includes all code lines of G6 and also the solid box except for the rounded box.

Game G8.

Initialization: $I_{\text{enc}} = I_\pi = I_\pi^{\mathcal{A}} = \emptyset$; $\mathsf{GenEnc}(1^k)$, $\text{pk}_R^*, \text{ID}_R^*$; $\text{IV}_1 = 0^r$; $\text{IV}_2 = 0^c$; $\text{IV}_3 = \text{IV}_2 \oplus 1$; $\mathcal{L}_c = \{\text{IV}_2, \text{IV}_3\}$

On decryption query K_h, Y_1, Y_2, T_k

1 **if** $\mathsf{Ver}_{\text{pk}_S}(Y_2 = (S^2, \sigma)) = \perp$ **then**
  return $\perp$
2 **if** there exists $\mathsf{checkin}(M, r, k, \ell_{\text{sg}}) = m_1 \| m_2 \| \cdots \| m_{n+1}$ such that, after setting $Y_1 = a_{e+1} \| \cdots \| a_j$, $u_{2_1} = \text{IV}_3, z_{1_1} = \text{IV}_1$ **then**
  if $\{(u_{1_i} \| u_{2_i}), (z_{1_i+1} \| z_{2_i+1})\} \in I_\pi^{\mathcal{A}}$ for $i = 1, \ldots, e, \ldots, j$ such that $a_i = u_{1_i} \oplus z_{1_i}, u_{2_i} = z_{2_i}$ and $\mathcal{O}^{\text{PC}}(S^1, Y_1) = 1$, where $S^1 = a_1 \| \cdots \| a_e$ **then**
    for setting $K = \lfloor z_j \rfloor_r \oplus K_h$, $S^1 \| S^2 = c_1 \| \cdots \| c_e \| c_{e+1} \| \cdots \| c_n$, $x_0 = K \| 0^{r-k} \oplus \text{IV}_1$, $T = T_k \oplus K$ and $w_0 = \text{IV}_2$
    if $(x_0 \| w_0, v_{1_1} \| v_{2_1}) \in I_\pi^{\mathcal{A}}$ and $\{(x_i \| w_i), (v_{1_{i+1}} \| v_{2_{i+1}})\} \in I_\pi^{\mathcal{A}}$ for $i = 1, \ldots, e, \ldots, n+1$ and $\lfloor v_{1_{n+2}} \rfloor_r = T$, where $x_i = c_i = m_i \oplus v_{1_i}, w_i = v_{2_i}$ **then**
      return M
    else
      return $\perp$
  else
    return $\perp$

The remaining oracles are the same as G7.

The following special notations are being used during G8 and onwards in the decryption oracle:

(i) During the SpongeWrap part of *SpPad*, we represent the input/output relation of π's subsequent calls for $pad(M) = m_1 \| \cdots \| m_n$ by $(v_{1_{i+1}} \| v_{2_{i+1}}) = \pi(x_i \| w_i)$, where $x_i = v_{1_i} \oplus \{m_i\}$, $w_i = v_{2_i}$, $0 \le i \le n$, $v_{1_0} = \text{IV}_1$, $m_0 = K$, $w_0 = \text{IV}_2$, $v_{1_i}, x_i \in \{0,1\}^r$ and $v_{2_i}, w_i \in \{0,1\}^c$. Then c_i will represent $m_i \oplus v_{1_i}$, where $1 \le i \le n$.

(ii) The input/output relation of π's subsequent call during the sponge part of *SpPad* will be represented by $(z_{1_i+1} \| z_{2_i+1}) = \pi(u_{1_i} \| u_{2_i})$, $u_{1_i} = c_i \oplus z_{1_i}$, $u_{2_i} = z_{2_i}$, where $1 \le i \le (j)$, $u_{2_1} = \text{IV}_3, z_{1_1} = \text{IV}_1, z_j = h$.

Figure 16: Game G8. Output of the decryption oracle in G8 is the same as G7 but independent from sk.

Games G9 and G10. In G9, K^* is generated randomly. In G10, K^* is computed using the value of randomly generated S^{1*}, K_h^* and Y_1^* . The value of K^* is calculated via $H^{\pi_{\text{enc}}}(\text{IV}_1 \| \text{IV}_2, y_1^*, y_2^*, \ldots, y_j^*) \oplus K_h^*$, where $y_1^*, y_2^*, \ldots, y_j^* = S^{1*} \| Y_1^*$. Here $H^{\pi_{\text{enc}}}(*)$ represents the sponge function with $\text{IV} = \text{IV}_1 \| \text{IV}_2$ using the permutation π_{enc}. Since π is an ideal permutation and K_h^* is a random value, K^* will also be random. Therefore, G9 and G10 are the same,

$$|\Pr[\text{Exp}_{\text{G10},\mathcal{A}}]| = |\Pr[\text{Exp}_{\text{G9},\mathcal{A}}]|.$$

Game G11. In G10, during signcryption $(K_h^*, S^{1*}, S^{2*}, T^*)$ was calculated using K^* and the r-bit random output of π. In G11, we directly allocate random K_h^*, S^{1*}, S^{2*}, T_k^* values to the signcryption oracle. Earlier, in G10, during signcryption, $(K_h^*, S^{1*}, S^{2*}, T_k^*)$ has a relation with K^*, whereas, in G11, there is no relation between $(K_h^*, S^{1*}, S^{2*}, T_k^*)$ and K^*. This gap can only be exploited if K^* is known to adversary $\mathcal{A}$ and queried $\text{ID}_S^* \| \text{ID}_R^* \| K^* \| \text{IV}_2$ to π. We mark this query by $\mathcal{A}$ to π as $bad_K \leftarrow true$. Therefore, $|\Pr[\text{Exp}_{\text{G11},\mathcal{A}}] - \Pr[\text{Exp}_{\text{G10},\mathcal{A}}]| \le \Pr[bad_K \leftarrow true]$. If this bad_K event does not happen, then essentially K_h^*, S^{1*}, S^{2*}, T_k^* will be random and also independent from M_d.

Game G12. This is the final game of adversary $\mathcal{A}$. It is the same as G11; if bad_K does not happen, then essentially S^{1*} remains unknown to the adversary along with K^*. The bad_K event in G11 is the same as $bad1_K$ in G12 because the sign-ciphertext is random and independent of M_d; therefore

$$|\Pr[\text{Exp}_{\text{G12},\mathcal{A}}]| = |\Pr[\text{Exp}_{\text{G11},\mathcal{A}}]| = \tfrac{1}{2}.$$

Games G9 and G10.

Initialization: $I_{\text{enc}} = I_\pi = I_\pi^{\mathcal{A}} = \emptyset$; $\mathsf{GenEnc}(1^k)$, pk_R^*, ID_R^*; $\text{IV}_1 = 0^r$; $\text{IV}_2 = 0^c$; $\text{IV}_3 = \text{IV}_2 \oplus 1$; $\mathcal{L}_c = \{\text{IV}_2, \text{IV}_3\}$

After find stage (AFS): $K_h^* \xleftarrow{\$} \{0,1\}^k$; $S^{1*} \xleftarrow{\$} \{0,1\}^\ell$; $S^{2*} \xleftarrow{\$} \{0,1\}^{\ell_{\text{sg}}}$
$Y_1^* = \mathsf{Enc}(S^{1*})$; $T^* \xleftarrow{\$} \{0,1\}^k$
$y_1 \| \cdots \| y_e \| y_{e+1} \| \cdots \| y_j = S^{1*} \| Y_1^*$
$K^* \xleftarrow{\$} \{0,1\}^k$

$K^* \| * = \pi_{\text{enc}}(\ldots \pi_{\text{enc}}(\pi_{\text{enc}}(y_1 \oplus \text{IV}_1 \| \text{IV}_3) \oplus y_2 \| 0^c) \cdots \oplus y_j \| 0^c) \oplus K_h^* \| 0^{b-k}$

On encryption query (M_d)

1 $K^* \xleftarrow{\$} \{0,1\}^k$; $x = \text{IV}_1 = \text{ID}_S^* \| \text{ID}_R^* \| 0^k$; $w = \text{IV}_2$
2 $\mathsf{checkin}(M, r, k, \ell_{\text{sg}}) = m_1 \| \cdots \| m_{(n+1)}$
3 $x = \text{ID}_S^* \| \text{ID}_R^* \| K$
4 **for** $i = 1, \ldots, (n+1)$ **do**
  $(x \| w) = \pi_{\text{enc}}(x \| w)$; $x = x \oplus m_i$; $c_i^* = x$
5 $(x \| w) = \pi_{\text{enc}}(x \| w)$; $T^* = \lfloor x \rfloor_k$
6 $S^{1*} \| S^{2*} \| T^* = c_1^* \| \cdots \| c_e^* \| c_{e+1}^* \| \cdots \| c_{n+1}^* \| T^*$
7 $Y_1^* = \mathsf{Enc}_{\text{pk}_R^*}(S^1)$; $\sigma^* = \mathsf{Sign}_{\text{sk}_S^*}(S^{2*})$
8 $pad(S^{1*} \| Y_1^*) = y_1^* \| \cdots \| y_j^*$; $x = \text{IV}_1$; $w = \text{IV}_3$
9 **for** $i = 1, \ldots, j$ **do**
  $(x \| w) = \pi_{\text{enc}}(x \oplus y_i^* \| w)$
10 $K_h^* = \lfloor x \rfloor_k \oplus K^*$; $T_k^* = T \oplus K$
11 **return** $(K_h^*, Y_1^*, Y_2^* = (S^{2*}, \sigma^*), T_k^*)$

The remaining oracles are the same as G8.

Figure 17: Games G9 and G10. G9 includes some extra dummy variables, shown in the dashed box, during initialization after find stage. G10 includes the solid-box code during initialization, in which K^* is chosen from random C^*.

Game G11.

Initialization: $I_{\text{enc}} = I_\pi = I_\pi^{\mathcal{A}} = \emptyset$; $\mathsf{GenEnc}(1^k)$, pk_R^*, ID_R^*; $\text{IV}_1 = 0^r$; $\text{IV}_2 = 0^c$; $\text{IV}_3 = \text{IV}_2 \oplus 1$; $\mathcal{L}_c = \{\text{IV}_2, \text{IV}_3\}$
AFS: $K_h^* \xleftarrow{\$} \{0,1\}^k$; $S^{1*} \xleftarrow{\$} \{0,1\}^\ell$; $S^{2*} \xleftarrow{\$} \{0,1\}^{\ell_{\text{sg}}}$
$Y_1^* = \mathsf{Enc}(S^{1*})$; $\sigma^* = \mathsf{Sign}_{\text{sk}_S^*}(S^{2*})$; $T_k^* \xleftarrow{\$} \{0,1\}^k$
$y_1 \| \cdots \| y_e \| y_{e+1} \| \cdots \| y_j = S^{1*} \| Y_1^*$
$K^* \| * = \pi_{\text{enc}}(\ldots \pi_{\text{enc}}(\pi_{\text{enc}}(y_1 \oplus \text{IV}_1 \| \text{IV}_3) \oplus y_2 \| 0^c) \cdots \oplus y_j \| 0^c) \oplus K_h^* \| 0^{b-k}$

On encryption query (M_d)

1 **return** $(K_h^*, Y_1^*, Y_2^* = (S^{2*}, \sigma^*), T_k^*)$

On $\pi_{\mathcal{A}}$-query m

1 **if** $(m = \text{ID}_S^* \| \text{ID}_R^* \| K^* \| \text{IV}_2)$ **then**
  $bad_K \leftarrow true$
2 $v = \pi(m)$
3 $I_\pi^{\mathcal{A}} = I_\pi^{\mathcal{A}} \cup \{(m, v)\}$
4 **return** v

On $\pi_{\mathcal{A}}^{-1}$-query v

1 $m = \pi^{-1}(v)$
2 **if** $(m = \text{ID}_S^* \| \text{ID}_R^* \| K^* \| \text{IV}_2)$ **then**
  $bad_K \leftarrow true$
3 $I_\pi^{\mathcal{A}} = I_\pi^{\mathcal{A}} \cup \{(m, v)\}$
4 **return** v

The remaining oracles are the same as G10.

Figure 18: Game G11. All values of the encryption oracle are replaced by random variables if the adversary does not query K^* to $\pi_{\mathcal{A}}$.

Game G12.

Initialization: $I_{\text{enc}} = I_\pi = I_\pi^{\mathcal{A}} = \emptyset$; $\text{GenEnc}(1^k)$, pk_R^*, ID_R^*; $\text{IV}_1 = 0^r$; $\text{IV}_2 = 0^c$; $\text{IV}_3 = \text{IV}_2 \oplus 1$; $\mathcal{L}_c = \{\text{IV}_2, \text{IV}_3\}$

AFS: $K_h^* \xleftarrow{\$} \{0,1\}^k$; $S^{1*} \xleftarrow{\$} \{0,1\}^\ell$; $S^{2*} \xleftarrow{\$} \{0,1\}^{\ell_{\text{sg}}}$

$Y_1^* = \text{Enc}_{\text{pk}}(S^{1*})$; $\sigma^* = \text{Sign}_{\text{sk}_S^*}(S^{2*})$; $T_k^* \xleftarrow{\$} \{0,1\}^k$

$y_1 \| \cdots \| y_e \| y_{e+1} \| \cdots \| y_j = S^{1*} \| Y_1^*$;

$K^* \| * = \pi_{\text{enc}}(\ldots \pi_{\text{enc}}(\pi_{\text{enc}}(y_1 \oplus \text{IV}_1 \| \text{IV}_3) \oplus y_2 \| 0^c) \cdots \oplus y_j \| 0^c) \oplus K_h^* \| 0^{b-k}$

On encryption query (M_d)

1 **return** $(K_h^*, Y_1^*, Y_2^* = (S^{2*}, \sigma^*), T_k^*)$

On $\pi_{\mathcal{A}}$-query m

1 **if** $(m = \text{ID}_S^* \| \text{ID}_R^* \| K^* \| \text{IV}_2)$ **then**
  $bad_K \leftarrow true$
2 $v = \pi(m)$
3 $I_\pi^{\mathcal{A}} = I_\pi^{\mathcal{A}} \cup \{(m, v)\}$
4 **return** v

On $\pi_{\mathcal{A}}^{-1}$-query v

1 $m = \pi^{-1}(v)$
2 **if** $(m = \text{ID}_S^* \| \text{ID}_R^* \| K^* \| \text{IV}_2)$ **then**
  $bad_K \leftarrow true$
3 $I_\pi^{\mathcal{A}} = I_\pi^{\mathcal{A}} \cup \{(m, v)\}$
4 **return** v

The remaining oracles are the same as G11.

Red color shows parts that are not detectable by the adversary.

Adversary C. Given random $Y_1 \xleftarrow{\$} \{0,1\}^{\ell + \text{co}_{\text{pe}}}$, find S^1 such that $\mathcal{O}^{\text{PC}}(S^1, Y_1) = 1$.

Game G12 as Adversary A.

Initialization: $I_{\text{enc}} = I_\pi = I_\pi^{\mathcal{A}} = \emptyset$, $\text{GenEnc}(1^k)$, pk_R^*, ID_R^*; $\text{IV}_1 = 0^r$; $\text{IV}_2 = 0^c$; $\text{IV}_3 = \text{IV}_2 \oplus 1$; $\mathcal{L}_c = \{\text{IV}_2, \text{IV}_3\}$

AFS: $K_h^* \xleftarrow{\$} \{0,1\}^k$; $\sigma^* = \text{Sign}_{\text{sk}_S^*}(S^{2*})$; $T_k^* \xleftarrow{\$} \{0,1\}^k$; $Y_1^* = Y_1$

The remaining oracles are the same as G12.

Finalization:

if $\{(u_{1_i} \| u_{2_i}), (z_{1_{i+1}} \| z_{2_{i+1}})\} \in I_\pi^{\mathcal{A}}$ for $i = 1, \ldots, e, \ldots, j$ such that $a_i = u_{1_i} \oplus z_{1_i}$, $u_{2_i} = z_{2_i}$ and $\mathcal{O}^{\text{PC}}(S^1, Y_1) = 1$, where $S^1 = a_1 \| \cdots \| a_e$, $Y_1 = a_{e+1} \| \cdots \| a_j$, $u_{2_1} = \text{IV}_3$ and $z_{1_1} = \text{IV}_1$ **then**
  return S^1

Figure 19: Game G12 as final game, and adversary C using G12 as adversary A

The probability of $bad1_K$ is

$$\begin{aligned}\Pr[bad1_K] &= \Pr[\text{ID}_S^* \| \text{ID}_R^* \| K^* \| \text{IV}_2 \text{ is queried to } (\pi_{\mathcal{A}} \text{ or } \pi_{\mathcal{A}}^{-1})] \\ &= \Pr[(\text{ID}_S^* \| \text{ID}_R^* \| K^* \| \text{IV}_2 \text{ is queried to } (\pi_{\mathcal{A}} \text{ or } \pi_{\mathcal{A}}^{-1})) \wedge (I_{\text{enc}} \subset I_\pi^{\mathcal{A}})] \\ &\quad + \Pr[(\text{ID}_S^* \| \text{ID}_R^* \| K^* \| \text{IV}_2 \text{ is queried to } (\pi_{\mathcal{A}} \text{ or } \pi_{\mathcal{A}}^{-1})) \wedge (I_{\text{enc}} \not\subset I_\pi^{\mathcal{A}})],\end{aligned}$$

where $(I_{\text{enc}} \subset I_\pi^{\mathcal{A}})$ implies that all the input/output relations of π_{enc} are also known to the adversary $\mathcal{A}$ via the set $I_\pi^{\mathcal{A}}$. Therefore, $\mathcal{A}$ knows all y_i^* for $1 \le i \le e$ and h^*. Moreover, the adversary $\mathcal{A}$ learns K^* from K_h^* of the challenged ciphertext.

Given $(K_h^*, Y_1^*, Y_2^* = (S^{2*}, \sigma^*), T^*)$, if $\text{ID}_S^* \| \text{ID}_R^* \| K^* \| \text{IV}_2$ is queried to π, then it reveals S^{1*} completely. Therefore,

$$\Pr[bad2] \le \text{Adv}_{\text{ENCRYPT}}^{\text{OW-PCA}}(\mathcal{B}_{\mathcal{A}}) + \Pr[(K^* \| \text{IV}_2 \text{ is queried to } (\pi_{\mathcal{A}} \text{ or } \pi_{\mathcal{A}}^{-1})) \wedge (I_{\text{enc}} \not\subset I_\pi^{\mathcal{A}})],$$

where $I_{\text{enc}} \not\subset I_\pi^{\mathcal{A}}$ implies that one of the inputs to $H^{\pi_{\text{enc}}}()$ is unknown to the adversary $\mathcal{A}$. This results in an unknown output value from $H^{\pi_{\text{enc}}}()$. Since K_h^* is already random, therefore K^* remains unknown and random

to $\mathcal{A}$. Since ID^* and IV_2 are public, the query $\mathrm{ID}_S^* \parallel \mathrm{ID}_R^* \parallel K^* \parallel \mathrm{IV}_2$ to $\pi_{\mathcal{A}}$ is equivalent to random guessing of K^*.

$$\Pr[bad2] \leq \mathrm{Adv}^{\text{OW-PCA}}_{\text{ENCRYPT}}(\mathcal{B}_{\mathcal{A}}) + \frac{(q_{\pi_{\mathcal{A}}} + q_{\pi^{-1}})}{\min(2^k, 2^c)}.$$

The last game G12 can be used to simulate adversary $\mathcal{B}$ for simulating adversary $\mathcal{A}$'s queries. Here the adversary tries to recover the first k bits from the input to Enc on given random y and other public information. □

Funding: Xavier Boyen is supported by Australian Research Council Future Fellowship grant FT140101145. Josef Pieprzyk was supported by Polish National Science Centre grant DEC-2014/15/B/ST6/05130 and Australian Research Council Discovery grant DP180102199.

References

[1] M. Abe, R. Gennaro and K. Kurosawa, Tag-KEM/DEM: A new framework for hybrid encryption, *J. Cryptology* **21** (2008), no. 1, 97–130.
[2] J. H. An, Y. Dodis and T. Rabin, On the security of joint signature and encryption, in: *Advances in Cryptology – EUROCRYPT 2002*, Lecture Notes in Comput. Sci. 2332, Springer, Berlin (2002), 83–107.
[3] C. Badertscher, F. Banfi and U. Maurer, A constructive perspective on signcryption security, in: *Security and Cryptography for Networks – SCN 2018* Lecture Notes in Comput. Sci. 11035, Springer, Berlin (2018), 102–120.
[4] J. Baek, R. Steinfeld and Y. Zheng, Formal proofs for the security of signcryption, in: *Public Key Cryptography – PKC 2002* Lecture Notes in Comput. Sci. 2274, Springer, Berlin (2002), 80–98.
[5] J. Baek, W. Susilo, J. K. Liu and J. Zhou, A new variant of the Cramer–Shoup KEM secure against chosen ciphertext attack, in: *Applied Cryptography and Network Security – ACNS 2009* Lecture Notes in Comput. Sci. 5536, Springer, Berlin (2009), 143–155.
[6] T. K. Bansal, D. Chang and S. K. Sanadhya, Sponge based CCA2 secure asymmetric encryption for arbitrary length message, in: *Information Security and Privacy – ACISP 2015* Lecture Notes in Comput. Sci. 9144, Springer, Berlin (2015), 93–106.
[7] M. Bellare and P. Rogaway, Optimal asymmetric encryption, in: *Advances in Cryptology – EUROCRYPT 1994*, Lecture Notes in Comput. Sci. 950, Springer, Berlin (1995), 92–111.
[8] M. Bellare and P. Rogaway, The exact security of digital signatures - how to sign with RSA and rabin, in: *Advances in Cryptology – EUROCRYPT 1996*, Lecture Notes in Comput. Sci. 1070, Springer, Berlin (1996), 399–416.
[9] M. Bellare and P. Rogaway, Code-based game-playing proofs and the security of triple encryption, preprint (2004), http://eprint.iacr.org/2004/331.
[10] M. Bellare and P. Rogaway, The security of triple encryption and a framework for code-based game-playing proofs, in: *Advances in Cryptology – EUROCRYPT 2006*, Lecture Notes in Comput. Sci. 4004, Springer, Berlin (2006), 409–426.
[11] G. Bertoni, J. Daemen, M. Peeters and G. V. Assche, Duplexing the sponge: Single-pass authenticated encryption and other applications, in: *Selected Areas in Cryptography – SAC 2011*, Lecture Notes in Comput. Sci. 7118, Springer, Berlin (2011), 320–337.
[12] G. Bertoni, J. Daemen, M. Peeters and G. V. Assche, Permutation-based encryption, authentication and authenticated encryption, preprint (2012).
[13] G. Bertoni, J. Daemen, M. Peeters and G. V. Assche, Keccak, in: *Advances in Cryptology – EUROCRYPT 2013*, Lecture Notes in Comput. Sci. 7881, Springer, Berlin (2013), 313–314.
[14] T. E. Bjørstad and A. W. Dent, Building better signcryption schemes with tag-kems, in: *Public Key Cryptography – PKC 2006*, Lecture Notes in Comput. Sci. 3958, Springer, Berlin (2006), 491–507.
[15] T. E. Bjørstad, A. W. Dent and N. P. Smart, Efficient KEMs with partial message recovery, in: *Cryptography and Coding*, Lecture Notes in Comput. Sci. 4887, Springer, Berlin (2007), 233–256.
[16] D. Chiba, T. Matsuda, J. C. N. Schuldt and K. Matsuura, Efficient generic constructions of signcryption with insider security in the multi-user setting, in: *Applied Cryptography and Network Security – ACNS 2011*, Lecture Notes in Comput. Sci. 6715, Springer, Berlin (2011), 220–237.
[17] R. Cramer and V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack, in: *Advances in Cryptology – Crypto 1998*, Lecture Notes in Comput. Sci. 1462, Springer, Berlin (1998), 13–25.
[18] A. W. Dent, A designer's guide to KEMs, in: *Cryptography and Coding*, Lecture Notes in Comput. Sci. 2898, Springer, Berlin (2003), 133–151.
[19] A. W. Dent, Hybrid signcryption schemes with insider security, in: *Information Security and Privacy – ACISP 2005*, Lecture Notes in Comput. Sci. 3574, Springer, Berlin (2005), 253–266.

[20] A. W. Dent, Hybrid signcryption schemes with outsider security, in: *Information Security – ISC 2005*, Lecture Notes in Comput. Sci. 3650, Springer, Berlin (2005), 203–217.
[21] A. W. Dent and Y. Zheng, *Practical Signcryption*, Springer, Berlin, 2010.
[22] Y. Dodis, M. J. Freedman, S. Jarecki and S. Walfish, Versatile padding schemes for joint signature and encryption, in: *Proceedings of the 11th ACM Conference on Computer and Communications Security – CCS'04*, ACM, New York (2004), 344–353.
[23] Y. Dodis, M. J. Freedman and S. Walfish, Parallel signcryption with oaep, pss-r, and other feistel paddings, preprint (2003), http://eprint.iacr.org/2003/043.
[24] E. Fujisaki and T. Okamoto, Secure integration of asymmetric and symmetric encryption schemes, *J. Cryptology* **26** (2013), no. 1, 80–101.
[25] T. E. Gamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Inform. Theory* **31** (1985), no. 4, 469–472.
[26] M. P. Guido Bertoni, Joan Daemen and G. V. Assche, Sponge functions, ECRYPT Hash Function Workshop, 2007.
[27] E. Kiltz, Chosen-ciphertext security from tag-based encryption, in: *Theory of Cryptography – TCC 2006*, Lecture Notes in Comput. Sci. 3876, Springer, Berlin (2006), 581–600.
[28] K. Kurosawa and Y. Desmedt, A new paradigm of hybrid encryption scheme, in: *Advances in Cryptology – CRYPTO 2004*, Lecture Notes in Comput. Sci. 3152, Springer, Berlin (2004), 426–442.
[29] B. Libert and J. Quisquater, Efficient signcryption with key privacy from gap diffie-hellman groups, in: *Public Key Cryptography – PKC 2004*, Lecture Notes in Comput. Sci. 2947, Springer, Berlin (2004), 187–200.
[30] J. Malone-Lee and W. Mao, Two birds one stone: Signcryption using RSA, in: *Topics in Cryptology – CT-RSA 2003*, Lecture Notes in Comput. Sci. 2612, Springer, Berlin (2003), 211–225.
[31] T. Matsuda, K. Matsuura and J. C. N. Schuldt, Efficient constructions of signcryption schemes and signcryption composability, in: *Progress in Cryptology – INDOCRYPT 2009*, Lecture Notes in Comput. Sci. 5922, Springer, Berlin (2009), 321–342.
[32] T. Okamoto and D. Pointcheval, REACT: Rapid enhanced-security asymmetric cryptosystem transform, in: *Topics in Cryptology – CT-RSA 2001*, Lecture Notes in Comput. Sci. 2020, Springer, Berlin (2001), 159–175.
[33] J. Pieprzyk and D. Pointcheval, Parallel authentication and public-key encryption, in: *Information Security and Privacy – ACISP 2003*, Lecture Notes in Comput. Sci. 2727, Springer, Berlin (2003), 387–401.
[34] J. Pieprzyk and D. Pointcheval, Parallel signcryption, in: *Practical Signcryption*, Springer, Berlin (2010), 175–192.
[35] V. Shoup, OAEP reconsidered, *J. Cryptology* **15** (2002), no. 4, 223–249.
[36] R. Steinfeld and Y. Zheng, A signcryption scheme based on integer factorization, in: *Information Security – ISW 2000*, Lecture Notes in Comput. Sci. 1975, Springer, Berlin (2000), 308–322.
[37] C. H. Tan, Signcryption scheme in multi-user setting without random oracles, in: *Advances in Information and Computer Security – IWSEC 2008*, Lecture Notes in Comput. Sci. 5312, Springer, Berlin (2008), 64–82.
[38] Y. Zheng, Digital signcryption or how to achieve cost(signature & encryption)<<cost(signature) + cost(encryption), in: *Advances in Cryptology – CRYPTO 1997*, Lecture Notes in Comput. Sci. 1294, Springer, Berlin (1997), 165–179.
[39] SHA3 Hash function competition, 2007; http://csrc.nist.gov/groups/ST/hash/sha-3/index.html, last visited 02-Jan-2017.