Formalizing the Relationship Between Commitment and Basic Cryptographic Primitives

S. Sree Vivek[1*†], S. S,harmila Deva Selvi[2*], Pallavi Chandrasekar[3‡], and C. Pandu Rangan[4]
[1]Pfizer Device R&D, Chennai, India
ssree.vivek@pfizer.com
[2]Microsoft Research, Bangalore, India
a-ssdeva@microsoft.com, sharmioshin@gmail.com
[3]Morgan Stanley, Bangalore, India
ccpallavi@gmail.com
[4]IIT Madras, Chennai, India
prangan@cse.iitm.ac.in

Abstract

Signcryption is a cryptographic primitive which offers the functionality of both digital signature and encryption with lower combined computational cost. On the other hand, commitment scheme allows an entity to commit to a value, where the entity reveals the committed value later during a decommit phase. In this paper, we explore the connection between commitment schemes, public key encryption, digital signatures and signcryption. We establish formal relationship between commitment and the other primitives. Our main result is that we show signcryption can be used as a commitment scheme with appropriate security notions. We show that if the underlying signcryption scheme is IND-CCA2 secure, then the hiding property of the commitment scheme is satisfied. Similarly, we show that if the underlying signcryption scheme is unforgeable, then the relaxed biding property of the commitment scheme is satisfied. Moreover, we prove that if the underlying signcryption scheme is NM-CCA2, then the commitment scheme is non-malleable.

Keywords: Commitment schemes, Signcryption, Encryption, Digital signature, Chosen-ciphertext attack, Generic construction, Non-Malleable Commitment.

1 Introduction

Gilles Brassard, David Chaum and Claude Crepeau [5] in 1988 formalized the concept of *Commitment Schemes*. A commitment scheme is a cryptographic primitive which allows one to commit to a value while at the same time keeping the value hidden, with the ability to reveal the committed value later. The notion of commitment schemes can be best explained with the following motivating example from [8] given by Ivan Damgård. If one wants to commit to a bit b, he writes the bit in a paper, places the paper in a box and locks the box. The locked box is now sent to the receiver. At the time of de-commitment, the sender provides the key corresponding to the lock to the receiver. The receiver could unlock the box and reveal the bit committed by the sender. During this process, the sender should not be able to change the value committed after he sends the commitment, to the receiver. This crucial property is called as *binding*. The locked box should not reveal any information about the bit committed to the receiver.

Journal of Internet Services and Information Security (JISIS), volume: 6, number: 4 (November 2016), pp. 35-53

*Work done when the author was a student at IIT Madras

†Corresponding author: No 51, Beach Home Avenue, II Street, Besant Nagar, Chennai-600090, Tel: +919894619816, Web: `http://sreevivek.page.tl`

‡Work done when the author was a student intern at IIT Madras

This property is called as *hiding*. Any commitment scheme should possess both these properties. Non-malleable commitments [10] are schemes where it is computationally hard for an adversary to produce a commitment on a message after seeing a commitment on a related message, and to decommit the commitment made by him into a valid message, on viewing the decommitment of the commitment seen earlier. Commitment scheme have wide applications like zero-knowledge proofs([15]), secure multi-party computation([14]), contract signing and many more. It can also be directly used in remote bidding. It is also used as building block in some cryptographic schemes such as authenticated encryption and secret sharing.

In 1997, Yuliang Zheng [21] proposed a public-key cryptographic primitive named *Signcryption* which fulfills the functionalities of digital signature and encryption in a single logical step, and with a cost significantly smaller than that required by signature followed by encryption or any other composition of the both. Signcryption provides both message confidentiality and authenticity.

The connection between cryptographic primitives plays a very important role. Earlier, M. Choudary Gorantla et al. established the connection between Signcryption and One-pass Key Establishment in [16]. On the other hand, Jian Weng et al., in [20], gave a formal relation between Identity-Based Proxy Re-Encryption and Mediated Identity-Based Encryption. Yael Gertner et al. formalized the connection between Public Key Encryption and Oblivious Transfer in [13]. Crescenzo et al. in [7] have mentioned the use of encryption in commitment schemes. Yet another motivation to work on black-box schemes is to enhance the efficiency, bring out abstract properties, prove lower bounds and impossibilities [18]. We take the study one step further in our paper by providing black-box constructions from well-known protocols rather than from abstract function classes. This has advantages of efficiency and extendibility of a reliable and stable implementation of one protocol to achieve other functionality. We explore the connection between commitment schemes, public key encryption, digital signatures and signcryption.

In this paper, we formalize the relationship between commitment schemes and other cryptographic primitives like encryption, signature and signcryption, i.e., it can be seen that encryption, signature or signcryption scheme can be used to construct commitment schemes. Usually in commitment schemes, the commitment is produced by manipulating the message and some random values. Generally, the private key and the public key of either the sender $\mathscr{S}$ or the receiver $\mathscr{R}$ is not used in commitment schemes. While using other primitives as commitment scheme, the sender generates the key pairs $\langle (sk_s, pk_s), (sk_r, pk_r) \rangle$ by invoking the key generation algorithm `KeyGen` twice. The commitment is produced according to the underlying primitive used and the keys generated by the sender. After receiving the decommitment, the receiver decommits the commitment to interpret the message given to him initially.

Research on commitment schemes are being done ever since 1988 ([5]). Ivan Damgård has provided survey of some new and old results on the existence of commitment schemes and its connection between zero-knowledge protocols in [8]. Shai Halevi et al. in [17] have proposed a practical string-commitment scheme which is provably secure based solely on collision-free hashing. It enables a computationally bounded party to commit strings to an unbounded one, and is optimal in terms of interaction, communication, and computation. Moreover, many non-malleable commitment schemes have been proposed. In [6], non-malleable commitment without any interaction and based on any one-way function was proposed by Crescenzo. An effective non-interactive non-malleable commitment scheme in the public parameter model, based on discrete logarithms and RSA assumptions were proposed by Cresenzo et al. in [7]. Ivan Damgård et al. have constructed non-interactive non-malleable commitments in [9], that remain non-malleable even if the adversary has access to an arbitrary number of commitments from honest players - rather than one, as in several previous schemes. In [11], DOU Jiawei and LI Shundong have proposed three new efficient non-malleable commitment schemes which does not require the a sender to prove that he knows his committed secret by zero-knowledge proof.

2 Preliminaries

In this section, we discuss the two communication models for commitment schemes, provide the formal definition and security models for commitment, signature, encryption and signcryption scheme.

2.1 Communication models

Commitment schemes can be designed in one of the following two communication models.

1. *Public-Parameter Model* - In this model ([3]) both the sender and receiver of the commitment share a public reference string which is an output of an efficient PPT algorithm.

2. *Public-random string model* - In this model ([12]) both the sender and the receiver share a public reference string which is assumed to be uniformly distributed.

In all our definitions we follow the public-parameter model. The definitions for the public-random string model can be easily obtained by replacing the algorithm generating the public reference string with an algorithm which picks the strings uniformly at random.

2.2 Commitment Scheme

2.2.1 Framework for Commitment Scheme:

A commitment scheme in public-parameter model consists of the following three algorithms (`Setup`, `Commit`, `Decommit`):

`Setup`(1^k): The setup algorithm takes input 1^k, where k is the security parameter. It generates the public reference string $PPparams$ for the public-parameter model. We write $PPparams \leftarrow$ `Setup`(1^k).

`Commit`$(PPparams, m)$: The commit algorithm takes a message m, where $m \in \mathscr{M}$, the message space as input. It outputs a pair (com, dec) for m. The component com serves as the commitment value, and dec serves as the decommitment value. We write $(com, dec) \leftarrow$ `Commit`$(PPparams, m)$.

`Decommit`$(PPparams, com, dec)$: The decommit algorithm takes the public reference string $PPparams$, the commitment com and the decommitment dec as input. If the decommitment is valid, it outputs the committed message m; otherwise it outputs $\perp$. We write $m \leftarrow$ `Decommit`$(PPparams, com, dec)$.

Correctness: For any message $m \in \mathscr{M}$,

$$\texttt{Decommit}(PPparams, \texttt{Commit}(PPparams, m)) = m$$

2.2.2 Security Model:

In the introduction we have informally stated the two security requirements for commitment schemes, namely *hiding* and *binding*. Now, we provide the formal definitions for both these properties.

Hiding: This property guarantees that the commitment scheme reveals no information about the message to the receiver. That is, it is computationally hard for any adversary $\mathscr{A}$ to generate two messages $m_0, m_1 \in \mathscr{M}$, such that given a target commitment com, $\mathscr{A}$ can distinguish whether it is a commitment of m_0 or m_1. For any PPT algorithm $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$, we require:

$$\begin{aligned} Pr[\ & PPparams \leftarrow \texttt{Setup}(1^k);\ (m_0, m_1, \alpha) \leftarrow \mathscr{A}_1(PPparams); \\ & b \varepsilon_R \{0,1\}; (com, dec) \leftarrow \texttt{Commit}(PPparams, m_b); \\ & b' \leftarrow \mathscr{A}_2(com, \alpha)\ :\ b = b'\ ] \end{aligned} \quad \leq \frac{1}{2} + negl(k)$$

Binding: The sender cannot decommit a committed value *com* in two different ways. That is, it is computationally hard for any adversary $\mathscr{A}$ to come up with (com, dec, dec'), such that (com, dec) and (com, dec') are valid commitments for messages m and m' such that $m \neq m'$. For any PPT $\mathscr{A}$, we require:

$$Pr[\, PPparams \leftarrow \texttt{Setup}(1^k);\ (com, dec, dec') \leftarrow \mathscr{A}(PPparams);\\ m \leftarrow \texttt{Decommit}(PPparams, com, dec);\\ m' \leftarrow \texttt{Decommit}(PPparams, com, dec')\ :\\ m \neq m' \wedge m, m' \neq \perp] \leq negl(k)$$

2.2.3 Relaxed commitment scheme:

In *relaxed* commitment scheme, the binding property of the regular commitment schemes is replaced by the **Relaxed Binding** property. It is computationally hard for an adversary $\mathscr{A}$ to give a message m, such that when the sender generates a valid commitment and decommit pair $(com, dec) \leftarrow \texttt{Commit}(PPparams, m)$, $\mathscr{A}(com, dec, PPparams)$ produces another decommit value dec' which is a valid decommitment to some $m' \neq m$ with non-negligible advantage. For any PPT $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$, we require:

$$Pr[\, PPparams \leftarrow \texttt{Setup}(1^k);\ (m, \alpha) \leftarrow \mathscr{A}_1(PPparams);\\ (com, dec) \leftarrow \texttt{Commit}(PPparams, m);\ dec' \leftarrow \mathscr{A}_2(com, dec; \alpha);\ \leq negl(k)\\ m' \leftarrow \texttt{Decommit}(PPparams, com, dec') :\ m \neq m' \wedge m, m' \neq \perp]$$

2.2.4 Non-malleable commitment scheme:

According to the definition of Dolev et al. [10], a commitment scheme is non-malleable if it is computationally hard for an adversary to generate a commitment com' for a message m' on seeing a commitment *com* for a related message m. Schemes satisfying the definition of [10] are called *non-malleable commitment schemes*. However, in the definition of [6], a commitment scheme is *non-malleable with respect to opening*, if it is computationally hard for the adversary to decommitment the modified commitment on seeing the decommitment of the original message. For non-malleability, two games are considered.

In the first game, a tuple of messages is generated according to some distribution. An adversary receives the commitment to these messages and generates a tuple of commitments himself. After receiving the decommitments of the original commitments, the adversary tries to decommit his commitments in a way that his messages are related to the original messages. The adversary wins if the messages are related. In this game, the adversary $\mathscr{A}$ is a probabilistic polynomial time interactive Turing machine. It learns *PPparams*, M and z_M. In its first invocation $\mathscr{A}$ receives a tuple of commitments $\overrightarrow{com}$ to the messages in $\overrightarrow{m}$. It responds with $\overrightarrow{com'}$. The adversary $\mathscr{A}$ should not directly copy any of the commitments in $\overrightarrow{com}$ into $\overrightarrow{com'}$. Later $\mathscr{A}$ is activated again, this time receiving a tuple $\overrightarrow{dec}$ of decommitments to the commitments in $\overrightarrow{com}$. It must now try to produce a tuple of decommitments $\overrightarrow{dec'}$ to its own commitments.

In the second game, a tuple of messages is generate according to the same distribution. This time, the commitment to these messages are not given to the adversary. The adversary has to generate a tuple of related messages without knowing anything about the original messages. In this game, a modified key generator $\widetilde{\texttt{Setup}}$ is used to generate the public key *pk*. This key must be indistinguishable from a real key. In addition to the key *pk*, the modified key generator also produces some extra information s_{pk} about the key. Let the adversary be named as $\mathbb{B}$.

In both the games there is a message generator M. The message generator receives the public reference string *PPparams*, of the commitment scheme and it also gets some auxiliary input z_M. M returns a vector of messages $\overrightarrow{m}$ and a value s. The value s contains information from M possibly including the

public key pk and the auxiliary input. In both the games the adversaries receive the description of M. The time complexity of M is bounded by some polynomial in k.

The adversary in the second game $\mathscr{B}$ is only invoked once. Like $\mathscr{A}$, it learns pk, M and z_M. It also gets s_{pk} as input. However, it will not receive any information about the tuple of messages $\overrightarrow{m}$ except for the number of messages in the tuple, $t = |\overrightarrow{m}|$. It returns a tuple $\overrightarrow{m'}$ of messages from $\mathscr{M} \cup \{\perp\}$.

The outcome of these two games could be compared by running a polynomial time distinguisher D. This distinguisher gets as input $s, \overrightarrow{m}, \overrightarrow{m'}$ and z_D, where in the first game $\overrightarrow{m'}$ is the resulting vector of messages when running the decommitment algorithm on the tuple $\overrightarrow{com'}$ and $\overrightarrow{dec'}$. The distinguisher returns a single bit, where 1 is interpreted as being a success. The probability of D outputting 1 cannot be increased by changing a message in $\overrightarrow{m}$ to $\perp$. This way the adversary $\mathscr{A}$ cannot get an advantage by deliberately refusing to open its commitments.

In the first game, success probability is given by

$$\begin{aligned}&Succ_{\mathscr{A},M,D}(k, z_M, z_D) = \\ &Pr[PPparams \leftarrow \texttt{Setup}(1^k);\ (s, \overrightarrow{m}) \leftarrow M(PPparams, z_M);\\ &(\overrightarrow{com}, \overrightarrow{dec}) \leftarrow \texttt{Commit}(PPparams, \overrightarrow{m});\ \overrightarrow{com'} \leftarrow \mathscr{A}(PPparams, \overrightarrow{com}, M, z_M);\\ &\overrightarrow{dec'} \leftarrow \mathscr{A}(\overrightarrow{dec}); \overrightarrow{m'} \leftarrow \texttt{Decommit}(PPparams, \overrightarrow{com'}, \overrightarrow{dec'}) : D(s, \overrightarrow{m}, \overrightarrow{m'}, z_D) = 1]\end{aligned}$$

In the second game, success probability is given by

$$\begin{aligned}&\widehat{Succ}_{\mathscr{B},M,D}(k, z_M, z_D) = \\ &\quad Pr[pk, s_{pk} \leftarrow \widehat{\texttt{Setup}}(1^k); (s, \overrightarrow{m}) \leftarrow M(pk, z_M);\\ &\quad (\overrightarrow{m'}) \leftarrow \mathscr{B}(pk, s_{pk}, t, M, z_M)\ :\ D(s, \overrightarrow{m}, \overrightarrow{m'}, z_D) = 1]\end{aligned}$$

Definition. A commitment scheme is non-malleable if it has a modified key generator $\widehat{\texttt{Setup}}$ such that for all $\mathscr{A}$ there exists a $\mathscr{B}$, where for all M and D we have $Succ_{\mathscr{A},M,D}(k, z_M, z_D) - \widehat{Succ}_{\mathscr{B},M,D}(k, z_M, z_D) < negl(k)$ for all z_M, z_D with lengths bounded by some polynomial in k.

ε-Non-malleable commitment scheme. A commitment scheme is ε-non-malleable if for all $\varepsilon > 0$ and every probabilistic polynomial time algorithm $\mathscr{A}$, there exists a simulator $\mathscr{B}$ running in polynomial time in k and ε^{-1} such that $Succ_{\mathscr{A},M,D}(k, z_M, z_D) - \widehat{Succ}_{\mathscr{B},M,D}(k, z_M, z_D) \leq \varepsilon + negl(k)$

The formal definition and security notions for digital signatures and public key encryption schemes could be found in [19] and hence omitted due to want of space.

2.3 Signcryption

2.3.1 Specification of the Cryptosystem:

A signcryption scheme consists of three algorithms : `Setup, Signcrypt, Unsigncrypt`.

`Setup`(1^k) : The `Setup` algorithm takes input 1^k, where k is the security parameter. It generates the public parameters δ for the public-parameter model. We write $\delta \leftarrow \texttt{Setup}(1^k)$.

`Keygen`(1^k): The `Keygen` algorithm takes input 1^k and generates the private key sk_r and the public key pk_r of the user. We write $(sk_s, pk_s) \leftarrow \texttt{Keygen}(1^k)$ to generate the sender's keys and $(sk_r, pk_r) \leftarrow \texttt{Keygen}(1^k)$ to generate the receiver's keys.

`Signcrypt`($m, sk_s,\ pk_s,\ pk_r, \delta$): The `Signcrypt` algorithm takes input the message m, the sender's secret key sk_s and the receiver's public key pk_r, and returns the ciphertext σ. We write $\sigma \leftarrow \texttt{Signcrypt}(m, sk_s, pk_r, \delta)$.

`Unsigncrypt`(σ, sk_r, pk_s, δ): The `Unsigncrypt` algorithm takes input the ciphertext σ, the receiver's secret key sk_r and sender's public key pk_s. If the signature is invalid, $\perp$ is returned, else message is returned unambiguously. We write m or $\perp \leftarrow$ `Unsigncrypt`($\sigma, sk_r, pk_s, \delta$).

Correctness: Any signcryption scheme should be correctly verifiable.

2.3.2 Security model:

In defining security for signcryption, we distinguish between two classes of attacks: *outsider attacks*, in which the attacker has access only to public information, and *insider attacks*, in which the attacker additionally has either the sender's or the recipient's private key. In each case, we must ensure that the two basic security goals of signcryption, *confidentiality* and *authenticity*, are provided. Any signcryption scheme should have the following properties, and these properties are formally defined in [4], and their formal proofs are proposed in [1] and [4].

- *Message confidentiality* : allows the communicating parties to preserve the secrecy of their exchange, if they choose to.
- *Ciphertext authentication* : allows the legitimate recipient alone, to be convinced that the ciphertext and the signed message it contains were crafted by the same entity. This implies ciphertext integrity. It also reassures the recipient that the communication was indeed secured end-to-end.

2.3.3 IND-CCA2 Security:

Let $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$ be a polynomial time adversary who is trying to break the indistinguishability property of the IND-CCA2 scheme, *SC*. The adversary is provided with a training phase where the adversary can get access to the signcryption and unsigncryption oracles. After the training phase, when given the public parameters, the adversary outputs two messages m_0 and m_1. A value b $\in_R$ {0,1} is chosen and the challenge ciphertext C^* of m_b is given to the adversary. The adversary is again provided with the access to signcryption and unsigncryption oracles with a restriction that he should not query the challenge ciphertext to the unsigncryption oracle. After the second phase of training, the adversary has to distinguish whether the C^* is the ciphertext of m_0 or m_1. If SC(.) and USC(.) are the signcryption and unsigncryption oracles, the advantage of the adversary is given by

$$Adv^{IND-CCA2}_{\mathscr{A},SC}(k) = 2.Pr[\delta \leftarrow \texttt{Setup}(1^k);\ (sk_s, pk_s) \leftarrow \texttt{Keygen}(1^k);$$
$$(sk_r, pk_r) \leftarrow \texttt{Keygen}(1^k);\ (m_0, m_1, s) \leftarrow \mathscr{A}_1^{SC(.),USC(.)}(\delta); b \in_R \{0,1\}$$
$$C^* \leftarrow \texttt{Signcrypt}(m_b, pk_r, sk_s); b' \leftarrow \mathscr{A}_2^{SC(.),USC(.)}(s, m_0, m_1, C^*) : b' = b] - 1$$

IND-iCCA2 is the insider secure IND-CCA2 scheme, where it is computationally hard for the receiver to attack the integrity of the scheme. Since the attacker is the either the sender or receiver, he knows the secret key of one of the communicators. IND-oCCA2 is the outsider secure IND-CCA2 scheme, where it is computationally hard for an entity, who is neither the sender nor the receiver, to attack the integrity of the scheme. Hence, the attacker has no knowledge about the keys of the sender and the receiver.

2.3.4 NM-CCA2 Security:

It formalizes an adversary's inability to find a ciphertext C', given a ciphertext C^*,such that the underlying plaintexts m' and m are related in some way. Given a signcryption scheme *SC* = (`Setup, KeyGen, Signcrypt, Unsigncrypt`), an adversary $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$ be a probabilistic polynomial-time algorithm.

For b $\in_R \{0,1\}$, signcryption oracle SC(.) and unsigncryption oracle USC(.), let $\mathsf{Exp}^{atk-b}_{\mathscr{A},SC}$ be the following game.

- $\delta \leftarrow \mathtt{Setup}(1^k)$
- $(sk_s, pk_s) \leftarrow \mathtt{Keygen}(1^k)$
- $(sk_r, pk_r) \leftarrow \mathtt{Keygen}(1^k)$
- $(M, s) \leftarrow \mathscr{A}_1{}^{USC(.)}(\delta)$;
- $m \xleftarrow{R} M$
- $C^* \leftarrow \mathtt{Signcrypt}(m, sk_s, pk_r)$
- $(R, \overrightarrow{C}) \leftarrow \mathscr{A}_2{}^{SC(.),USC(.)}(M, s, C^*)$
- $\overrightarrow{m} \leftarrow \mathtt{Unsigncrypt}(\overrightarrow{C}, pk_s, sk_r)$
- *return* 1 if $(C^* \notin \overrightarrow{C}) \wedge (\perp \notin \overrightarrow{m}) \wedge R(m, \overrightarrow{m})$ *else return* 0.

The advantage of the adversary id given by

$$Adv^{NM-CCA2}_{\mathscr{A},SC}(k) \;=\; Pr[\mathsf{Exp}^{atk-1}_{\mathscr{A},SC}(k)] \;-\; Pr[\mathsf{Exp}^{atk-0}_{\mathscr{A},SC}(k)]$$

2.3.5 UF-CMA Security:

To break the UF-CMA security of the signcryption scheme, adversary $\mathscr{A}$ has to come up with a valid signcryption C^* of a new message $m*$, for which it did not ask the sender to signcrypt earlier and for which the unsigncryption oracle does not return $\perp$. Note that, $\mathscr{A}$ is not required to know the message when producing the ciphertext, although $\mathscr{A}$ can always compute the message by querying the unsigncryption oracle with the ciphertext. If SC(.) is the signcryption oracle,

- $\delta \leftarrow \mathtt{Setup}(1^k)$
- $(sk_s, pk_s) \leftarrow \mathtt{Keygen}(1^k)$
- $(sk_r, pk_r) \leftarrow \mathtt{Keygen}(1^k)$
- $(C^*, m^*) \leftarrow \mathscr{A}^{SC(.),USC(.)}(\delta, pk_s, pk_r)$
- $\mathscr{A}$ wins if m^* has not been queried to the signcryption oracle and if C^* is a valid ciphertext of message m^*

UF-iCMA is the insider secure UF-CMA scheme, where it is computationally hard for the receiver to attack the integrity of the scheme. Since either the sender or the receiver attacks the signcryption scheme, he has an upper hand in knowing the public and private key of one of the communicating entities. UF-oCMA is the outsider secure UF-CMA scheme, where it is computationally hard for an entity, who is neither the sender nor the receiver, to attack the integrity of the scheme. Hence, the attacker has no knowledge about the keys of the sender and the receiver.

3 Connecting the Primitives

In the following sections, the `KeyGen` function used to generate the secret key / public key pair is implemented as shown below.

`Setup`: $PPparams \leftarrow \texttt{Setup}(1^k)$

`KeyGen`: *Choose sk randomly*, $pk \leftarrow \texttt{PublicKeyGen}(PPparams, sk)$ and $Output = \langle sk, pk \rangle$.

Here, `PublicKeyGen` generates the public key for the selected secret key *sk* based on a hard problem.

3.1 Using Signcryption as Commitment scheme

A signcryption scheme with `Setup`, `KeyGen`, `Signcrypt` and `Unsigncrypt` algorithms is assumed to be available. Let $\mathscr{S}$ be the sender and $\mathscr{R}$ be the receiver of the commitment scheme. Assume that $\mathscr{S}$ works with the signcryption scheme with the above mentioned algorithms. The public parameters δ generated by `Setup` algorithm of signcryption scheme, is used as public reference string $\triangle\_PPparams$ in commitment scheme. $\mathscr{S}$ and $\mathscr{R}$ share only the public reference string $\triangle\_PPparams$ in common. The Commitment scheme $\triangle$ is modeled as follows.

`Setup`(1^k): $\triangle\_PPparams = \delta \leftarrow \texttt{Setup}(1^k)$

`Commit`(m, $\triangle\_PPparams$): To commit a message m, $\mathscr{S}$ does the following to generate commitment and the corresponding decommitment.

1. Invoke `KeyGen` twice to generate two key pairs $\langle (sk_s, pk_s), (sk_r, pk_r) \rangle$. Formally,
   - $(sk_s, pk_s) \leftarrow \texttt{KeyGen}(\triangle\_PPparams)$
   - $(sk_r, pk_r) \leftarrow \texttt{KeyGen}(\triangle\_PPparams)$
2. Generate the ciphertext using `Signcrypt` algorithm of the signcryption scheme and the key values generated in previous step.
   - $\sigma \leftarrow \texttt{Signcrypt}(m, sk_s, pk_s, pk_r)$
   - $\triangle\_com = \langle \sigma, pk_s, pk_r \rangle$
   - $\triangle\_dec = sk_r$
   - $Output \langle \triangle\_com, \triangle\_dec \rangle$

After generating the commitment $\triangle\_com$ and decommitment $\triangle\_dec$, the sender $\mathscr{S}$ sends only the commitment $\triangle\_com$ to the receiver $\mathscr{R}$. At a later point of time, to reveal the message, the sender $\mathscr{S}$ sends $\triangle\_dec = sk_r$ to $\mathscr{R}$. $\mathscr{R}$ apprehends the message in decommitment phase by using `Unsigncrypt` algorithm of the signcryption scheme.

`Decommit`($\triangle\_PPparams$, $\triangle\_com$, $\triangle\_dec$)

- $m \leftarrow \texttt{Unsigncrypt}(\triangle\_com = \langle \sigma, pk_s, pk_r \rangle, \triangle\_dec = sk_r)$
- *Output m*

The *verification* algorithm pertaining to the Signcryption scheme is executed in decommitment phase.

NOTE.

For each and every commitment to be generated, the sender $\mathscr{S}$ generates fresh key pairs $\langle (sk_s, pk_s), (sk_r, pk_r) \rangle$.

3.1.1 Properties for Commitment scheme:

The three important properties of commitment schemes are, Hiding, Binding and Non-malleability

If the underlying signcryption scheme is IND-CCA2 (the ciphertext doesn't give any information of the message inside), the commitment scheme exhibits hiding property (the commitment doesn't give any information about the message inside), as the ciphertext (along with the public keys) in signcryption is used as commitment in the commitment scheme. Similarly, it has been proved in [2] that a scheme which is IND-CCA2 secure, is NM-CCA2 secure too. Thus, signcryption scheme of IND-CCA2 security is non-malleable. We have proved that when a non-malleable signcryption is used as a commitment scheme, the non-malleability property of commitment scheme holds. Hence, an IND-CCA2 secure signcryption scheme provides both hiding and non-malleability properties of commitment schemes. On the other hand, we have proved that if the underlying signcryption scheme is UF-CMA, the relaxed binding property of the commitment scheme holds. The formal proof for the above statements are given below.

Theorem 3.1. ($IND-CCA2 \implies Hiding$) *The commitment scheme $\triangle$ exhibits hiding property, if the underlying signcryption scheme is IND-CCA2 secure.*

It can be proved that if the hiding property of the commitment scheme is broken, then the signcryption scheme is no longer IND-CCA2 secure. Consider,

- $\mathscr{A}$ - Adversary of the commitment scheme
- $\mathscr{C}$ - Adversary of the signcryption scheme and challenger to $\mathscr{A}$
- $\mathscr{B}$ - Challenger of the signcryption scheme

Consider a signcryption scheme which is IND-CCA2 secure and let $\mathscr{C}$ be the adversary of the signcryption scheme intending to break the IND-CCA2 security of the scheme. Assume that adversary $\mathscr{A}$ can break the hiding property of the commitment scheme. $\mathscr{C}$ acts as the challenger of the commitment scheme, who makes use of $\mathscr{A}$'s ability. Thus, if the hiding property of commitment scheme is broken by its adversary $\mathscr{A}$, the adversary of signcryption scheme $\mathscr{C}$ breaks the IND-CCA2 security of the signcryption scheme using $\mathscr{A}$. This leads to contradiction as the considered signcryption scheme is IND-CCA2 secure. Hence, if the underlying signcryption scheme is IND-CCA2 secure, then the commitment scheme exhibits hiding property. **Fig.1** explains how an IND-CCA2 secure signcryption scheme provides hiding property, when used as a commitment scheme.

`Setup` algorithm of the signcryption scheme is run by $\mathscr{B}$ to generate the public parameters δ and sends it to $\mathscr{C}$. $\mathscr{C}$ being the challenger of the commitment schemes, sends those public parameters $\triangle\_PP\_params = \delta$ to $\mathscr{A}$. Thus, all the entities use the common reference string, $\triangle\_PPparams$.

$$\triangle\_PPparams = \delta \leftarrow \texttt{Setup}(1^k)$$

The formal game between the three entities $\mathscr{A}$, $\mathscr{B}$ and $\mathscr{C}$, is shown below.

- $\mathscr{C}$ generates: $(sk_s, pk_s) \leftarrow \texttt{Keygen}(1^k)$ and $(sk_r, pk_r) \leftarrow \texttt{Keygen}(1^k)$
- $\mathscr{A}$ to $\mathscr{C}$: $(m_0, m_1, \alpha) \leftarrow \mathscr{A}(\triangle\_PPparams)$
- $\mathscr{C}$ to $\mathscr{B}$: $(m_0, m_1) \leftarrow \mathscr{C}^{USC(.),SC(.)}(m_0, m_1, sk_s, pk_s)$
- $\mathscr{B}$ generates: $b\ \varepsilon_R\ \{0,1\}$
- $\mathscr{B}$ performs: $C^* \leftarrow \texttt{Signcrypt}(m_b, pk_r, sk_s)$

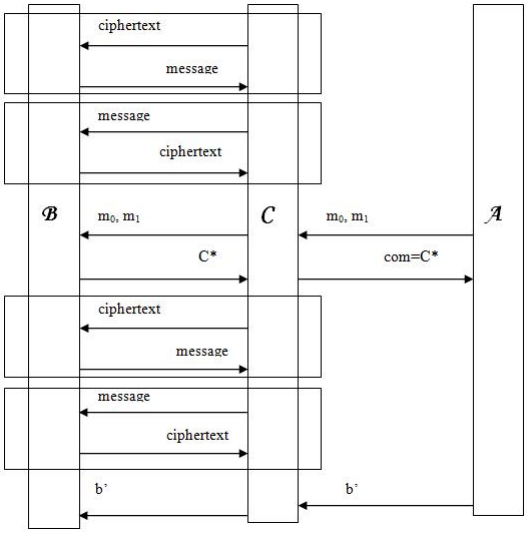


Figure 1: IND-CCA2 $\Longrightarrow$ Hiding

- $\mathscr{B}$ to $\mathscr{C}$ to $\mathscr{A}$: $com \leftarrow \mathscr{C}(C^*, pk_s, pk_r)$
- $\mathscr{A}$ to $\mathscr{C}$: $b' \leftarrow \mathscr{A}(com, \alpha)$
- $\mathscr{C}$ to $\mathscr{B}$: $b' \leftarrow \mathscr{C}^{USC(.),SC(.)}(b')$
- $\mathscr{B}$ *verifies if* $b \stackrel{?}{=} b'$

Recall that the key values $\langle (sk_s, pk_s), (sk_r, pk_r) \rangle$ are generated for each and every commitment to be produced. The adversary of the signcryption scheme $\mathscr{C}$ can undergo training phase by accessing the signcryption and unsigncryption oracles. The adversary of commitment scheme, $\mathscr{A}$ produces two messages (m_0, m_1) and sends them to its challenger $\mathscr{C}$. $\mathscr{C}$ passes on the messages m_0 and m_1 given by adversary of commitment scheme $\mathscr{A}$, to the challenger of the signcryption scheme $\mathscr{B}$. Also, $\mathscr{C}$ passes on the challenge ciphertext returned by $\mathscr{B}$ along with public keys, as commitment to $\mathscr{A}$. To break the hiding property, $\mathscr{A}$ produces b' and sends it to its challenger $\mathscr{C}$, which passes it to the challenger of signcryption scheme $\mathscr{B}$, intending to break the IND-CCA2 security of signcryption scheme. Thus, if the hiding property is broken, then signcryption scheme is IND-CCA2 insecure. So, if the underlying signcryption scheme is IND-CCA2 secure, then hiding property is exhibited in the commitment scheme derived from the signcryption scheme.

Theorem 3.2. (UF-CMA $\Longrightarrow$ Relaxed binding) *The commitment scheme $\triangle$ exhibits Relaxed binding property, if the underlying signcryption scheme is UF-CMA secure.*

According to relaxed binding property of the commitment scheme, given a commitment *com* and decommitment *dec* for a message *m* chosen by the adversary, it is computationally hard for the adversary to find decommitment dec' which decommits the commitment *com* to another message m'. A signcryption scheme is UF-CMA secure if it is computationally hard for an adversary to give a valid ciphertext *com* for a message *m*, for which it has not asked the signcryption oracle to signcrypt. Consider,

- $\mathscr{A}$ - Adversary of the commitment scheme
- $\mathscr{B}$ - Challenger of the signcryption scheme

- $\mathscr{C}$ - Adversary of the signcryption scheme and challenger to $\mathscr{A}$

Consider a signcryption scheme which is UF-CMA secure and let $\mathscr{C}$ be the adversary of the signcryption scheme intending to break the UF-CMA security of the scheme. Assume that adversary $\mathscr{A}$ can break the relaxed binding property of the commitment scheme. $\mathscr{C}$ acts as the challenger of the commitment scheme, who makes use of the adversary $\mathscr{A}$'s ability. Thus, if the relaxed binding property of commitment scheme is broken by its adversary $\mathscr{A}$, the adversary of signcryption scheme $\mathscr{C}$ breaks the UF-CMA security of the signcryption scheme using $\mathscr{A}$. This leads to contradiction as the considered signcryption scheme is UF-CMA secure. Hence, if the underlying signcryption scheme is UF-CMA secure the commitment scheme constructed from it exhibits relaxed binding property. **Fig.2** explains how an UF-CMA secure signcryption scheme provides relaxed binding property, when used as a commitment scheme.

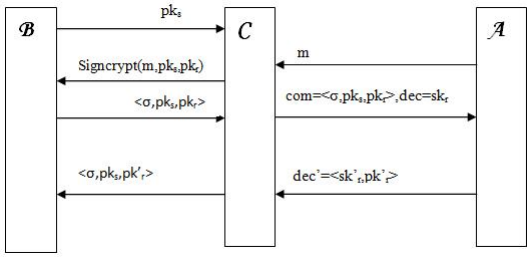


Figure 2: UF-CMA $\Longrightarrow$ Relaxed binding

`Setup` algorithm of the signcryption scheme is run by $\mathscr{B}$ to generate the public parameters δ and sends it to $\mathscr{C}$. $\mathscr{C}$ being the challenger of the commitment schemes, sends those public parameters $\triangle\_PPparams = \delta$ to $\mathscr{A}$. Thus, all the entities use the common reference string, $\triangle\_PPparams$.

$$\triangle\_PPparams = \delta \leftarrow \texttt{Setup}(1^k)$$

The formal game between the three entities $\mathscr{A}$, $\mathscr{B}$ and $\mathscr{C}$, is shown below.

- $\mathscr{C}$ generates: $(sk_s, pk_s) \leftarrow \texttt{Keygen}(1^k)$ and $(sk_r, pk_r) \leftarrow \texttt{Keygen}(1^k)$
- $\mathscr{A}$ to $\mathscr{C}$: $m \leftarrow \mathscr{A}(\triangle\_PPparams)$
- $\mathscr{C}$ to $\mathscr{B}$ to $\mathscr{C}$: $(\sigma, pk_s, pk_r) \leftarrow \texttt{Signcrypt}(m, pk_r, sk_s)$
- $\mathscr{C}$ to $\mathscr{A}$: $(com = \langle \sigma, pk_s, pk_r \rangle, dec = sk_r) \leftarrow \mathscr{C}(\sigma, pk_s, pk_r, sk_r)$
- $\mathscr{A}$ to $\mathscr{C}$: $dec' = \langle sk'_r, pk'_r \rangle \leftarrow \mathscr{A}(com, dec)$
- $\mathscr{C}$ to $\mathscr{B}$: $(\sigma, pk_s, pk'_r) \leftarrow \mathscr{C}(dec')$
- $\mathscr{B}$ *verifies if* σ *is a valid ciphertext for message m created using* $\langle sk'_r, pk'_r \rangle$

Recall that the key values $\langle (sk_s, pk_s), (sk_r, pk_r) \rangle$ are generated for each and every commitment to be produced. The adversary of the commitment scheme, $\mathscr{A}$ sends a message m to $\mathscr{C}$. $\mathscr{C}$, in turn, passes m to the challenger of the commitment scheme $\mathscr{B}$ along with key details of the receiver of signcryption scheme, pk_r and sk_r, which are generated by $\mathscr{C}$ using `KeyGen` algorithm. The challenger $\mathscr{B}$ accesses the signcryption oracle and gets the ciphertext σ for the message m using pk_r and sk_s. $\mathscr{B}$ sends $\langle \sigma, pk_s, pk_r \rangle$ to adversary $\mathscr{C}$, and $\mathscr{C}$ passes $com = \langle \sigma, pk_s, pk_r \rangle$ to $\mathscr{A}$ as commitment. The decommitment sent to $\mathscr{A}$ by $\mathscr{C}$ is sk_r. The adversary of the commitment scheme $\mathscr{A}$ finds another decommitment $dec' = \langle sk'_r, pk'_r \rangle$

and sends it to $\mathscr{C}$, breaking the relaxed binding property. $\mathscr{C}$, in turn, breaks the UF-CMA security of the signcryption scheme by sending $\langle \sigma, pk_s, pk'_r \rangle$ to the challenger of signcryption scheme $\mathscr{B}$, forging the signature as it maps to a different receiver with keys $\langle pk'_r, sk'_r \rangle$. So, if the relaxed binding property of the commitment scheme is broken, then the underlying signcryption scheme becomes UF-CMA insecure. Hence, if the underlying signcryption scheme is UF-CMA secure, then relaxed binding property is exhibited in the commitment scheme.

Theorem 3.3. (NM-CCA2 $\Longrightarrow$ Non-malleability) *The commitment scheme $\triangle$ exhibits non-malleability property, if the underlying signcryption scheme is NM-CCA2 secure.*

According to [10], a commitment scheme is non-malleable if it is computationally hard for an adversary to generate a commitment com' for a message m' on seeing a commitment com for a related message m. According to [6], a commitment scheme is *non-malleable with respect to opening* if it is computationally hard for the adversary to decommitment the modified commitment on seeing the decommitment of the original message. We have taken into account both these notions - non-malleability with respect to commitment and non-malleability with respect to decommitment (also known as non-malleability with respect to opening). Consider,

- $\mathscr{A}$ - Adversary of the commitment scheme
- $\mathscr{B}$ - Challenger of the signcryption scheme
- $\mathscr{C}$ - Adversary of the signcryption scheme and challenger to $\mathscr{A}$

Consider a signcryption scheme which is NM-CCA2 secure and let $\mathscr{C}$ be the adversary of the signcryption scheme intending to break the NM-CCA2 security of the scheme. Assume that adversary $\mathscr{A}$ can break the non-malleability property of the commitment scheme. The adversary of signcryption scheme acts as the challenger of the commitment scheme, who makes use of the adversary $\mathscr{A}$'s ability. Thus, if the non-malleability property of commitment scheme is broken by its adversary $\mathscr{A}$, the adversary of signcryption scheme $\mathscr{C}$ breaks the NM-CCA2 security of the signcryption scheme using $\mathscr{A}$. This leads to contradiction as the considered signcryption scheme is NM-CCA2 secure. Hence, if the underlying signcryption scheme is NM-CCA2 secure, then the commitment scheme is non-malleable. **Fig.3** explains how a NM-CCA2 secure signcryption scheme provides non-malleability property, when used as a commitment scheme.

`Setup` algorithm of the signcryption scheme is run by $\mathscr{B}$ to generate the public parameters δ and sends it to $\mathscr{C}$. $\mathscr{C}$ being the challenger of the commitment scheme, sends those public parameters $\triangle\_PPparams = \delta$ to $\mathscr{A}$. Thus, all the entities use the common reference string, $\triangle\_PPparams$.

$$\triangle\_PPparams \leftarrow \texttt{Setup}(1^k)$$

The formal game between the three entities-$\mathscr{A}, \mathscr{B}$ and $\mathscr{C}$, is shown below.

- $\mathscr{B}$ to $\mathscr{C}$: $C^* \leftarrow \mathscr{B}(\triangle\_PPparams)$
- $\mathscr{C}$ to $\mathscr{A}$: $com \leftarrow \mathscr{C}(C^*)$
- $\mathscr{A}$ to $\mathscr{C}$: $com' \leftarrow \mathscr{A}(com)$
- *Repeat until* $\langle com', dec' \rangle$ *is a valid commitment pair*
  - $\mathscr{C}$ to $\mathscr{A}$: $dec \leftarrow \mathscr{C}(com, \triangle\_PPparams)$

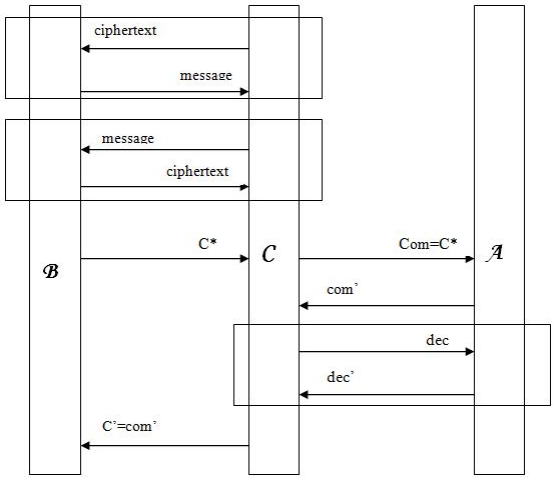


Figure 3: NM-CCA2 $\Longrightarrow$ Non-malleability

- $\mathscr{A}$ to $\mathscr{C}$: $dec' \leftarrow \mathscr{A}(com, com', dec)$

- $\mathscr{C}$ to $\mathscr{B}$: $C' \leftarrow \mathscr{C}(com', dec')$

- $\mathscr{B}$ *verifies if* C' *is a valid ciphertext*

Recall that the key values $\langle (sk_s, pk_s), (sk_r, pk_r) \rangle$ are generated for each and every commitment to be produced. The challenger of the signcryption scheme $\mathscr{B}$ provides training to the adversary of the signcryption scheme $\mathscr{C}$ by giving access to the signcryption and unsigncryption oracles. Once the training is over, the challenger of the signcryption scheme $\mathscr{B}$ provides the challenge ciphertext C^* to $\mathscr{C}$. $\mathscr{C}$ sends the challenge ciphertext, $com = C^*$ to adversary of commitment scheme $\mathscr{A}$ as a commitment. On seeing com, $\mathscr{A}$ produces a commitment com' to a related message (malleability). $\mathscr{C}$ produces a decommitment dec for the commitment com, such that it decommits to a valid message m. On seeing dec, $\mathscr{A}$ produces a decommitment dec' for the commitment com' given by him earlier. This continues until dec' decommits com' to a valid related message. When the valid decommitment is given, $\mathscr{C}$ passes $C' = com'$ as a valid ciphertext breaking the NM-CCA2 scheme. So, if the non-malleability property of the commitment scheme is broken, the underlying signcryption scheme becomes NM-CCA2 insecure. Hence, if the underlying signcryption scheme is NM-CCA2 secure, then the commitment scheme is non-malleable.

3.2 Using Signature as Commitment scheme

A signature scheme with `Setup, KeyGen, Sign` and `Verify` algorithms is assumed to be available. Let $\mathscr{S}$ be the sender and $\mathscr{R}$ be the receiver of the commitment scheme. The public parameters δ generated by `Setup` algorithm of signature scheme is used as public reference string $\gamma\_PPparams$ in commitment scheme. $\mathscr{S}$ and $\mathscr{R}$ share only the public reference string $\gamma\_PPparams$ in common. The commitment scheme γ is modeled as follows.

`Setup`(1^k): $\gamma\_PPparams = \delta \leftarrow$ `Setup`(1^k)

`Commit`(m, $\gamma\_PPparams$): To commit a message m, $\mathscr{S}$ does the following to generate commitment and decommitment.

1. Invokes `KeyGen` to generate the key values, sk_s and pk_s. Formally, $(sk_s, pk_s) \leftarrow$ `KeyGen`($\gamma\_PPparams$)

2. Generates the signature using `Sign` algorithm of the signature scheme and the key values generated in previous step. Compute $\sigma \leftarrow \texttt{Sign}(m \mid\mid pk_s, sk_s)$, $\gamma\_com = \sigma$, $\gamma\_dec = \langle m, pk_s \rangle$ and $Output \langle \gamma\_com, \gamma\_dec \rangle$

After generating the commitment γ_com and decommitment γ_dec, the sender $\mathscr{S}$ sends only the commitment γ_com to the receiver $\mathscr{R}$. At a later point in time, the sender $\mathscr{S}$ sends $\gamma\_dec = \langle m, pk_s \rangle$ to $\mathscr{R}$. On receiving the decommitment, $\mathscr{R}$ verifies the message m.

`Decommit`($\gamma\_PPparams$, γ_com, γ_dec): *Signature verification* is done in the decommitment phase. Given the commitment (signature) and the decommitment (message and public key), the given signature can be verified using `Verify` algorithm of the signature scheme.

NOTE. For each and every commitment to be generated, the sender $\mathscr{S}$ generates fresh keys sk_s and pk_s.

3.2.1 Hiding of the Commitment scheme:

According to the hiding property of the commitment schemes, given a commitment γ_com, the receiver should not be able to gain information about the message in the commitment. Applying the same here, while using a signature scheme as a commitment scheme, hiding property holds only if the receiver is not able to guess the message from a given signature. The commitment γ_com given to the receiver is the signature on $m \mid\mid pk_s$ using the secret key sk_s, generated using the `Sign` algorithm. The public key pk_s used for signature verification is given to the receiver only in the decommitment phase. Without the knowledge of the secret key and public key, the receiver cannot gain any information out of the commitment γ_com given to him during the commitment phase. Thus, the hiding property holds when a signature scheme is used as a commitment scheme.

	UF-CMA	sUF-CMA
Hiding	$\surd$	$\surd$

3.2.2 Binding of the Commitment scheme:

According to the binding property of the commitment schemes, a commitment γ_com should decommit only to one message m which is used in the commitment phase. In other words, it is computationally hard to decommit a commitment on message m to another message m'. While using a signature scheme as a commitment scheme, it should be computationally hard to relate a signature given to another message m'. By *integrity* property of signature schemes, if a message is signed, any change in the message after signature will invalidate the signature. That is, if a signature relates to another message m', it will fail during *signature verification* performed during the decommitment phase using `Verify` algorithm, and the commitment will be considered invalid. Thus, a commitment γ_com will decommit only to message m and the binding property holds.

3.3 Using Encryption as Commitment scheme

An encryption scheme with `Setup, KeyGen, Encrypt` and `Decrypt` algorithms is assumed to be available. Let $\mathscr{S}$ be the sender and $\mathscr{R}$ be the receiver of the commitment scheme. Assume that $\mathscr{S}$ works with the encryption scheme with the above mentioned algorithms. The public parameters δ generated by `Setup` algorithm of encryption scheme, is used as public reference string $\delta\_PPparams$ in commitment scheme. $\mathscr{S}$ and $\mathscr{R}$ share only the public reference string $\delta\_PPparams$ in common. The Commitment scheme δ is modeled as follows.

`Setup`(1^k): $\delta\_PPparams = \delta \leftarrow \texttt{Setup}(1^k)$

`Commit`(m, $\delta\_PPparams$): For each commitment of a message m, $\mathscr{S}$ does the following to generate commitment and decommitment.

1. Invokes `KeyGen` to generate the key values,sk_r and pk_r. Formally, $(sk_r, pk_r) \leftarrow$ `KeyGen`$(\delta\_PPparams)$
2. Generates the ciphertext using `Encrypt` algorithm of the encryption scheme and the key values generated in previous step. Compute $\sigma \leftarrow$ `Encrypt`(m, pk_r), $\delta\_com = \sigma$, $\delta\_dec = \langle m, sk_r \rangle$ and $Output \langle \delta\_com, \delta\_dec \rangle$

After generating the commitment δ_com and decommitment δ_dec, the sender $\mathscr{S}$ sends only the commitment δ_com to the receiver $\mathscr{R}$. At a later point in time, to reveal the message, the sender $\mathscr{S}$ sends $\delta\_dec = \langle m, sk_r \rangle$ to $\mathscr{R}$. $\mathscr{R}$ apprehends the message in decommitment phase, by using `Decrypt` algorithm of the encryption scheme.

`Decommit`($\delta\_PPparams$, δ_com, δ_dec): Compute $m' \leftarrow$ `Decrypt`(δ_com, δ_dec) and $Output\ m'$. Once the message is obtained on decryption, the following *verification* is done in the Decommitment phase.

- $pk'_r \leftarrow$ `PublicKeyGen`$(\delta\_PPparams, sk_r)$
- Check $pk'_r \stackrel{?}{=} pk_r$ and $m' \stackrel{?}{=} m$

NOTE.

For each and every commitment to be generated, the sender $\mathscr{S}$ generates the key values(sk_r and pk_r).

3.3.1 Hiding of the Commitment scheme

By the hiding property of the commitment schemes, given a commitment δ_com, the receiver should not be able to gain information about the message m in the commitment. While using an encryption scheme as a commitment scheme, hiding property holds only if the receiver is not able to guess the message from the given ciphertext. The commitment δ_com given to the receiver is nothing but the ciphertext of m created using `Encrypt` algorithm. It is computationally hard to gain knowledge of the message m from the given ciphertext δ_com without knowing the secret key sk_r, due to *confidentiality* property of the encryption schemes, i.e., without the knowledge of the secret key, the ciphertext cannot be apprehended. Thus, encryption schemes, when used as commitment schemes, provide hiding property.

3.3.2 Binding of the Commitment scheme

According to the binding property of the commitment schemes, it is computationally hard to decommit a commitment on message m to another message m'. While using an encryption scheme as a commitment scheme, binding property holds only if it is computationally hard to decrypt a ciphertext created using public key pk_r and message m, to another message m' using the corresponding secret key sk_r. By the *correctness* of the encryption schemes, for any message m,

$$\texttt{Decrypt}(sk_r, \texttt{Encrypt}(pk_r, m)) = m$$

Therefore, it can be seen that a commitment γ_com will decommit only to one message m and hence, the binding property holds. The following table shows the encryption schemes for which binding and non-malleability properties of commitment schemes hold good.

Encryption	Binding	Non-Malleability
CPA	√	X
CCA-1	√	X
CCA-2	√	√
NM-CCA	√	√
OW-CPA	√	X

4 Instantiating primitives

In this section, we take up a specific instance of signcryption scheme and show how the primitive can be used as a commitment scheme. To be used as a commitment scheme, outsider secure signcryption scheme is enough to provide security. For instance, consider the Zheng's scheme[21]. Let $\mathbb{G}$ be a prime order group for which extracting discrete logarithms is hard but multiplication is easy. Assume that p and q are prime numbers with $q|p-1$ and group $\mathbb{G} \subseteq \mathbb{Z}^*{}_p$. Let g be the generator of the group and x be randomly chosen from $\mathbb{Z}_q$. Let *PPparams* be the common reference string accepted by the sender and receiver at the time of instantiation. Assume *hash* to be a one-way hash function and *KH* to be keyed one-way hash function. According to the scheme, the ciphertext has three components-$\langle r, c, s \rangle$. Let $sk_s = s$ and $pk_s = g^s \ mod \ p$ be the secret key and public key of the sender. Let $sk_r = y$ and $pk_r = g^y \ mod \ p$ be the secret key and public key of the receiver.

Commitment phase:

- Generate $PPparams \leftarrow \texttt{Setup}(1^k)$
- Generate $(sk_s, pk_s) \leftarrow \texttt{Keygen}(PPparams)$ and $(sk_r, pk_r) \leftarrow \texttt{Keygen}(PPparams)$
- *Choose x randomly from* $\mathbb{Z}_q$, *i.e.*, $x \in_R \mathbb{Z}_q$
- $k = hash(pk_r{}^x \ mod \ p)$, *split k into* k_1 *and* k_2 *of appropriate length*
- Compute $r = KH_{k_2}(m, \ pk_r)$, $c = E_{k_1}(m)$, $s = x/(r + sk_s) \ mod \ q$, $com = \langle r, c, s, pk_s, pk_r \rangle$ and $dec = sk_r$

Sender sends $com = \langle r, c, s, pk_s, pk_r \rangle$ to the receiver as commitment.

Decommitment phase: Sender sends $dec = sk_r$ to the receiver as decommitment. The receiver computes $k = hash((pk_s \ g^r)^{s.sk_r} \ mod \ p)$, *splits k into* k_1 *and* k_2, computes $m = D_{k_1}(c)$ and *accepts m only if* $KH_{k_2}(m, pk_r) \stackrel{?}{=} r$.

The above underlying signcryption scheme is outsider secure. When used as a commitment scheme, it provides the hiding and binding property of the commitment schemes. Insider security notion cannot be provided as the sender generates the one-time key for the receiver.

5 Source authentication

We also extend our thought of using signcryption as commitment schemes, to source authentication. One of the major application of commitment schemes is bidding and auction. Source authentication is an important feature required during the bidding, i.e., only authorized users should bid. If one-time key is used by the sender, then source authentication cannot be provided as his key changes for every transaction. To provide source authentication, the key-pair of the sender should be fixed, i.e., the sender should use the same public key pk_s and secret key sk_s for all the transactions. According to our scheme,

when signcryption is used as a commitment scheme, sender's secret key sk_s is never revealed. Hence, if the sender's keys are permanent in the commitment scheme, he can be identified using his public key and can be authorized. The same concept can be used for spam filtering. To provide source authentication, a *publicly verifiable signcryption scheme* should be used as a commitment scheme. Since it is publicly verifiable, the sender can be identified even without knowing the message hidden inside the commitment.

6 Conclusion

We studied the formal relationship between commitment schemes and other cryptographic primitives such as signature, encryption and signcryption. We showed that these scheme when used as a commitment scheme provides hiding and binding property required for commitment schemes. In addition, we showed that if the underlying signcryption scheme is non-malleable, then the commitment scheme derived from it is also non-malleable. We have also discussed that a publicly verifiable signcryption scheme, when used as commitment scheme, provides source authentication.

References

[1] J. Baek, R. Steinfeld, and Y. Zheng. Formal proofs for the security of signcryption. *Journal of Cryptology*, 20(2):203–235, 2007.

[2] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Proc. of the 18th Annual International Cryptology Conference (CRYPTO'98), Santa Barbara, California, USA*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer Berlin Heidelberg, August 1998.

[3] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proc. of the 20th Annual ACM Symposium on Theory of Computing (STOC'88), Chicago, Illinois, USA*, pages 103–112. ACM, May 1988.

[4] X. Boyen. Multipurpose identity-based signcryption: A swiss army knife for identity-based cryptography. In *Proc. of the 23rd Annual International Cryptology Conference (Advances in Cryptology (CRYPTO'03), Santa Barbara, California, USA*, volume 2729 of *Lecture Notes in Computer Science*, pages 383–399. Springer Berlin Heidelberg, August 2003.

[5] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[6] G. D. Crescenzo, Y. Ishai, and R. Ostrovsky. Non-interactive and non-malleable commitment. In *Proc. of the 13th Annual ACM Symposium on the Theory of Computing (STOC'98), Dallas, Texas, USA*, pages 141–150. ACM, May 1998.

[7] G. D. Crescenzo, J. Katz, R. Ostrovsky, and A. Smith. Efficient and non-interactive non-malleable commitment. Cryptology ePrint Archive, Report 2001/032, 2001. `https://eprint.iacr.org/2001/032` [Online; Accessed on November 10, 2016].

[8] I. Damgård. Commitment schemes and zero-knowledge protocols. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark*, volume 1561 of *Lecture Notes in Computer Science*, pages 63–86. Springer Berlin Heidelberg, July 1998.

[9] I. Damgård and J. Groth. Non-interactive and reusable non-malleable commitment schemes. In *Proc. of the 35th Annual ACM Symposium on Theory of Computing (STOC'03), San Diego, California, USA*, pages 426–437. ACM, June 2003.

[10] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[11] J. Dou and S. Li. New efficient non-malleable commitment schemes. In *Proc. of the 6th International Conference on Advanced Language Processing and Web Information Technology (ALPIT'07), Luoyang, Henan, China*, pages 601–605. IEEE, August 2007.

[12] M. Fischlin and R. Fischlin. Efficient non-malleable commitment schemes. In *Proc. of the 20th Annual International Cryptology Conference (CRYPTO'00), Santa Barbara, California, USA*, volume 1880 of *Lecture Notes in Computer Science*, pages 413–431. Springer Berlin Heidelberg, August 2000.

[13] Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan. The relationship between public key encryption and oblivious transfer. In *Proc. of the 41st Annual Symposium on Foundations of Computer Science (FOCS'00), Redondo Beach, California, USA*, pages 325–335. IEEE, November 2000.

[14] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proc. of the 19th Annual ACM Symposium on Theory of Computing (STOC'87), New York, USA*, pages 218–229. ACM, May 1987.

[15] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity for all languages in np have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

[16] M. C. Gorantla, C. Boyd, and J. M. G. Nieto. On the connection between signcryption and one-pass key establishment. In *Proc. of the 11th IMA International Conference on Cryptography and Coding, Cirencester, UK*, volume 4887 of *Lecture Notes in Computer Science*, pages 277–301. Springer Berlin Heidelberg, December 2007.

[17] S. Halevi and S. Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *Proc. of the 16th Annual International Cryptology Conference (CRYPTO'96), Santa Barbara, California, USA*, volume 1109 of *Lecture Notes in Computer Science*, pages 201–215. Springer Berlin Heidelberg, August 1996.

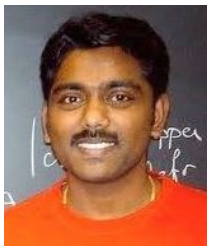
[18] O. Horvitz and J. Katz. Bounds on the efficiency of black-box commitment schemes. *Theoretical Computer Science*, 411(10):1251–1260, 2010.

[19] W. Mao. *Modern Cryptography: Theory and Practice*. Prentice Hall Professional Technical Reference, 2003.

[20] Z.-M. Wan, J. Weng, X.-J. Lai, S. Liu, and J. Li. On the relation between identity-based proxy re-encryption and mediated identity-based encryption. *Journal of Information Science and Engineering*, 27(1):243–259, 2011.

[21] Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) $<<$ cost(signature) + cost(encryption). In *Proc. of the 17th Annual International Cryptology Conference (CRYPTO'97), Santa Barbara, California, USA*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179. Springer Berlin Heidelberg, August 1997.

Author Biography



S.Sree Vivek has a PhD in Computer Science and Engineering from the Indian Institute of Technology - Madras (IIT-M), Chennai, India. He is currently working as Technical Architect (Cybersecurity Analysis) in Pfizer Device R&D, Chennai, India. His research interests include design and cryptanalysis of Cryptosystem, Embedded security, Mobile security and System security.

S.Sharmila Deva Selvi has a PhD in Computer Science and Engineering from the Indian Institute of Technology - Madras (IIT-M), Chennai, India. She works as Researcher in Microsoft Research, Bangalore, India. Her areas of interest are Provable Security of Public Key Cryptosystem, Cryptanalysis of Identity Based and Certificateless Cryptosystem, and designing solutions for secure storage and computation in the Cloud.

C Pallavi Chandrasekar received the B.E. in Computer Science and Engineering from The College of Engineering Guindy, Chennai. She is currently working as a Manager at Morgan Stanley, Bangalore, India. Her interests are Security, Algorithms and Cryptography.

C.Pandu Rangan is a Professor in the department of computer science and engineering of Indian Institute of Technology - Madras, Chennai, India. He obtained his PhD at the Indian Institute of Science (IISc), Bangalore in 1984. He heads the Theoretical Computer Science Lab in IIT Madras. His areas of interest are in theoretical computer science mainly focusing on Cryptography, Algorithms and Data Structures, Game Theory, Graph Theory and Distributed Computing.