Stronger Public Key Encryption Schemes Withstanding RAM Scraper Like Attacks

S. Sree Vivek, S. Sharmila Deva Selvi, C. Pandu Rangan

Theoretical Computer Science Lab,
Department of Computer Science and Engineering,
Indian Institute of Technology Madras, India.
{sharmila,svivek,prangan}@cse.iitm.ac.in

Abstract. Security of an encryption system is formally established through the properties of an abstract game played between a challenger and an adversary. During the game, the adversary will be provided with all information that he could obtain in an attack model so that the adversary is fully empowered to carry out the break. The information will be provided to the adversary through the answers of appropriately defined oracle queries. Thus, during the game, adversary will ask various oracle queries and obtain the related responses and have them at his disposal to effect a break. This kind of interaction between challenger and adversary is called as training to the adversary. For example, in the lunch time attack model, the adversary may ask encryption as well as decryption oracle queries. The indistinguishability of ciphertext under this model (IND-CCA2 model) is considered to offer strongest security for confidentiality. In the recent past, an adversary could obtain several additional information than what he could normally obtain in the CCA2 model, thanks to the availability of powerful malwares. In order to realistically model the threats posed by such malwares, we need to empower the adversary with answers to few other kinds of oracles. This paper initiates such a research to counter malwares such as RAM scrapers and extend the CCA2 model with additional oracles to capture the effect of RAM scrapers precisely. After discussing the new kind of attack/threat and the related oracle, we show that the transformation in [8] that yields a CCA2 secure system does not offer security against RAM scraper based attack. We refer the decryption oracle as glass box decryption oracle. We then propose two new schemes that offer security against glassbox decryption and also establish the formal security proof for the new schemes in random oracle and standard model.

Keywords: Public Key Encryption, Adaptive Chosen Ciphertext Secure with Glass Box Decryption, CPA-CCA2 Transformations, Random Oracle model, Standard model.

1 Introduction

The security notions for public key encryption systems have witnessed tremendous depth in their understanding over the past three decades. Currently, it is commonly agreed that an encryption scheme producing ciphertexts that are indistinguishable even by an adversary who carries an adaptive chosen ciphertext attack (CCA2) gives the best protection for the confidentiality of the encrypted message. The adversary who carries out the CCA2 attack has *black box* access to the encryption and decryption oracles. In other words, it is assumed that the adversary has only the input and outputs of various oracle queries and with these values, the adversary attempts to break the system. However, in real life, an adversary could obtain a number of additional related informations and these informations may help the adversary in some unexpected way to break the encryption scheme. The side channel attacks, for example obtains several information during the execution of oracles, other than the inputs and outputs of the queries. It is well known that these additional informations can be used effectively to break the cryptosystem that are proved secure in the traditional sense.

Thus it is important to extend the existing models and in the extended model, we must provide additional input/information to the adversary that he could obtain himself in real-life (besides the answers for encryption/decryption oracles). Such an adversary would model the real-life threats more accurately.

This paper addresses one such important extension to the CCA2 security model and accounts for additional information that an adversary would obtain. Our model is inspired by the following threat.

RAM Scraper is a piece of malware created to grab data residing in a systems volatile memory. RAM scrapers can be deployed to capture selective data than to effect bulk data grabs, thus avoiding dramatic increase in data traffic that could potentially raise *illicit traffic flag*. This is the key reason why operations of RAM scrapers never get noticed and the threats posed by RAM scrapers were added to the list of *Top Data Breach Attacks* by Verizon Business [3]. Since all the standards for secure communication of sensitive data requires end-to-end encryption while being transmitted, received or stored, the unencrypted information residing in RAM during decryption is a new hope for the adversary This is why the adversary targets the RAM precisely, using RAM scrapers. Now-a-days, a typical computing system that is running cryptographic algorithms will have a Trusted Platform Module (TPM) integrated in it. The private key of a user will be stored in TPM and the computations involving private keys will be carried out in TPM as showed in Fig. 1.

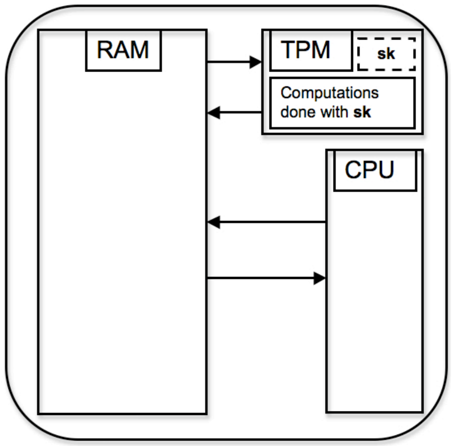

Fig. 1. Trusted Platform Module Deployed

The decryption algorithms and signing algorithms need the private key values for their computations. However, the private key values will not be *moved to* RAM. For example, the execution of a decryption algorithm may contain some computational steps that does not involve private key values and some steps using private keys values. The computations of first kind will be done in CPU and the resulting values will be available in RAM and the computations of the second type will be carried out in TPM after obtaining the necessary values from RAM. All the values obtained through the computations done in TPM will of-course be sent/available in RAM. If this computing system is affected by a RAM scraper malware, then the adversary may obtain all the intermediate values obtained during the execution but the adversary cannot obtain the private keys. The glass box decryption, introduced in this paper models this scenario exactly.

In view of the above scenario, we equip our adversary with a *Glass Box* decryption oracle rather than with the traditional *Black Box* decryption oracle. Specifically, we assume that, when the adversary makes a glass box decryption oracle query, he obtains, besides the output of the oracle, all the intermediate values generated during the execution of the decryption algorithm. We make the realistic assumption that the adversary does not obtain the private key in this process as discussed above since the steps involving the private key are carried out in trusted/secure memory but the resulting values are available in RAM.

Adaptive Chosen Ciphertext Security with Glass box Decryption: This model is stronger than CCA2 model in the sense that we have replaced the usual black box decryption oracle in CCA2 model with a stronger glass box oracle. All other oracles of CCA2 will also be considered in CCA2 model without any change. The glass box decryption oracle query, denoted by `Glass-Box-Dec`(c), returns the following:

– If the decryption algorithm is properly terminated with an output, then, the output and all the intermediate values generated during the execution will be returned.
– If the decryption algorithm is terminated with an $ABORT$, then the intermediate values generated up to that point will be returned.
– The private key values will not be returned.

Formally, we describe the values available to the adversary due to the execution of glass box decryption using the well-known notations of *Activation Records* and *Remote Procedure Calls*. Each execution of a procedure is carried out in the system with the help of a book keeping mechanism called *activation record*. Activation record consists of storages associated with variables declared in the procedure (parameters, local variables etc) [13]. Thus, we assume that all values available in the activation record is visible/available to the adversary. To model the scenario that all computations using secret keys are done in trusted platforms, we assume that all remote call procedures (specifically the calls made to trusted platform) are available only in a black box fashion. We also make one final reasonable assumption that the (remote) function calls made to the trusted platform are one-way functions of the secret key values stored in the trusted platform. In sum, if the decryption algorithm consists of several expression evaluation using the storages $S_1, S_2, \ldots$ and remote function calls $f_1, f_2, \ldots$ returning the values $F_1, F_2, \ldots$ respectively, then the execution of black box decryption makes $\langle S_1, S_2, \ldots, F_1, F_2, \ldots \rangle$ also to be available to the adversary.

1.1 Related Works:

Recently, few works have been reported in areas such as extending the security model and designing schemes against freezing attacks, where it is possible to measure a significant fraction of secret key bits if the secret key is stored in the part of memory which can be accessed even after power is turned off [1]. Their paper talks about retrieving the secret information from the system volatile memory. [9] deals with the issue of storing cryptographic keys and computing on them in a manner that preserves security even if the adversary obtains information through leakage during the computation on the key. The design of cryptographic primitives resilient to key-leakage attacks, where the attacker has the provision to repeatedly and adaptively learn information about the secret key, with the constraint that the overall amount of such information is bounded by some parameter was done in [2]. Recently, [11] talks about provable leakage resilient encryption schemes. It is to be noted that in all these works, resisting the side-channel attacks through which the attacker can retrieve the private key is being considered. Even though the schemes talk about mitigating the attacks due to key leakage, they do not consider the complete exposure of ephemeral keys. It should be further noted that even if encryption schemes are designed for key leakage resilience, RAM scrapers present a devastating threat to those systems too.

1.2 Our Contribution:

In this paper, we point out that the CCA2 notion (currently the strongest notion for confidentiality) will not capture the attack by RAM scrapers during the lunch time attack. We propose the new security notion for confidentiality, named as "Adaptive Chosen Ciphertext Attack Security with Glass box Decryption", which captures the attack by RAM scrapers. Many transformations are known till date for converting a CPA secure scheme to a CCA2 secure scheme in the random oracle model. We consider [8], the transformation which converts any CPA secure public key encryption scheme to a CCA2 secure public key encryption scheme. We propose two new stronger encryption scheme. Our first scheme is proved to be secure in the random oracle model and the second is proved in the standard model.

2 Preliminary Concepts

2.1 Computational Diffie Hellman Problem (CDH):

Definition 1. *Let κ be the security parameter and $\mathbb{G}$ be a multiplicative group of order q, where $|q| = \kappa$. Given $(g, g^a, g^b)_R \in \mathbb{G}$, the computational Diffie Hellman problem is to compute $g^{ab} \in \mathbb{G}$.*

The advantage of an adversary $\mathcal{A}$ in solving the computational Diffie Hellman problem is defined as the probability with which $\mathcal{A}$ solves the above computational Diffie Hellman problem.

$$Adv_{\mathcal{A}}^{CDH} = Pr[\mathcal{A}(g, g^a, g^b) = g^{ab}]$$

The computational Diffie Hellman assumption holds in $\mathbb{G}$ if for all polynomial time adversaries $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{CDH}$ is negligible.

2.2 Decisional Bilinear Diffie-Hellman Problem (DBDH):

Definition 2. *Given $(R, aR, bR, cR, \alpha) \in \mathbb{G}_1^4 \times \mathbb{G}_2$ for unknown $a, b, c \in \mathbb{Z}_q^*$, where $\mathbb{G}_1$ is a cyclic additive group with prime order q, the DBDH in $\langle \mathbb{G}_1, \mathbb{G}_2 \rangle$ is to decide whether $\alpha \overset{?}{=} \hat{e}(R, R)^{abc}$. Here, $\mathbb{G}_2$ is a multiplicative group with order q.*

The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving DBDH in $\langle \mathbb{G}_1, \mathbb{G}_2 \rangle$ is defined as

$$Adv_{\mathcal{A}}^{DBDH} = |Pr[\mathcal{A}(R, aR, bR, cR, \hat{e}(R, R)^{abc}) = 1] - Pr[\mathcal{A}(R, aR, bR, cR, \alpha) = 1]|$$

The DBDH Assumption is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{DBDH}$ is negligibly small.

2.3 CCA2 Security Notion:

- $\mathcal{C}$ generates $(sk, pk) \leftarrow \texttt{Gen}(\kappa)$ and pk is given to $\mathcal{A}$.
- $\mathcal{A}$ is given oracle access to $\texttt{Dec}_{sk}(.)$. $\mathcal{A}$ gives $\mathcal{C}$ two messages m_0 and m_1 of the same length.
- $\mathcal{C}$ chooses a random bit $\delta \leftarrow \{0, 1\}$ and generates the challenge ciphertext $c \leftarrow \texttt{Enc}_{pk}(m_\delta)$. and gives it to $\mathcal{A}$.
- $\mathcal{A}$ continues to get oracle access to $\texttt{Enc}_{pk}(.)$ as well as $\texttt{Dec}_{sk}(.)$ and outputs a bit δ' finally.
- $\mathcal{C}$ outputs 1 if $\delta = \delta'$ and 0 otherwise.

A public key encryption scheme $\mathcal{E} = (\texttt{Gen}, \texttt{Enc}, \texttt{Dec})$ has indistinguishable encryptions under adaptive chosen ciphertext attack (CCA2) if for all probabilistic polynomial time adversaries $\mathcal{A}$ the following advantage funtion is negligible in κ:

$$Adv_{\mathcal{A}}^{\mathcal{E},CCA2} = \left| Pr(b = b') - \frac{1}{2} \right|$$

2.4 CCA2 Security With Glass Box Decryption:

This model is identical to IND-CCA2 game, except that, `Glass-Box-Dec` oracle will be used by the adversary (instead of the regular black box decryption oracle).

The `Glass-Box-Dec` Oracle: Let $\mathcal{I}$ (denoting intermediate values) be the set of output due to the execution of $\texttt{Glass-Box-Dec}_{sk}(c)$ oracle with ciphertext c as input. This can be denoted as $\mathcal{I} \leftarrow \texttt{Glass-Box-Dec}_{sk}(c)$. $\mathcal{I}$ contains two different kinds of values:

– All values computed with the private key stored in the TPM. Note that the computations done with the private key in the TPM will be like a one-way function evaluation of the input and the private key. Hence it is not possible to retrieve the private key from the output value (e.g: Exponentiating the input with the private key in a Diffie-Hellman-based decryption scheme OR computing a Schnorr signature with a locally chosen (chosen inside the TPM) random number.)
– All other computations done outside the TPM, i.e inside the RAM.

So, the adversary virtually gets access to all computed values during decryption, except the private key. $\Diamond$

– $\mathcal{C}$ generates $(sk, pk) \leftarrow \texttt{Gen}(\kappa)$ and pk is given to $\mathcal{A}$.
– $\mathcal{A}$ is given oracle access to $\texttt{Glass-Box-Dec}_{sk}(.)$. $\mathcal{A}$ gives $\mathcal{C}$ two messages m_0 and m_1 of the same length.
– $\mathcal{C}$ chooses a random bit $\delta \leftarrow \{0, 1\}$ and generates the challenge ciphertext $c \leftarrow \texttt{Enc}_{pk}(m_\delta)$. and gives it to $\mathcal{A}$.
– $\mathcal{A}$ continues to get oracle access to $\texttt{Enc}_{pk}(.)$ as well as $\texttt{Glass-Box-Dec}_{sk}(.)$ and outputs a bit δ' finally.
– $\mathcal{C}$ outputs 1 if $\delta = \delta'$ and 0 otherwise.

A public key encryption scheme $\mathcal{E} = (\texttt{Gen}, \texttt{Enc}, \texttt{Dec})$ has indistinguishable encryptions under adaptive chosen ciphertext attack with glass box decryption if for all probabilistic polynomial time adversaries $\mathcal{A}$ the following advantage funtion is negligible in κ:

$$Adv_{\mathcal{A}}^{\mathcal{E},\texttt{GB}} = \left| Pr(b = b') - \frac{1}{2} \right|$$

3 Attack on the Fujisaki-Okamoto Transformation-1:

We use the plain CPA secure El-Gamal encryption scheme described below to instantiate the transformations.

Plain El-Gamal Encryption: The CPA secure El-Gamal encryption scheme $\mathcal{EL}$ is a tuple of probabilistic polynomial time algorithms ($\mathcal{EL}.\texttt{Gen}$, $\mathcal{EL}.\texttt{Enc}$, $\mathcal{EL}.\texttt{Dec}$) such that:

– $\mathcal{EL}.\texttt{Gen}$: The private key and public key pair of a user is $(sk, pk) = (x, g^x)$.
– $\mathcal{EL}.\texttt{Enc}$: Compute $c_1 = g^r$ and $c_2 = pk^r \oplus m$ and the ciphertext is $c = \langle c_1, c_2 \rangle$.
– $\mathcal{EL}.\texttt{Dec}$: To decrypt, compute $m = c_2 \oplus c_1^{sk}$.

3.1 Applying Fujisaki-Okamoto Transformation-1:

When we apply the Fujisaki-Okamoto Transformation-1 [8] to the plain El-Gamal encryption $\mathcal{EL}$, we obtain a CCA2 secure public key encryption scheme, which is described below:
CCA2 Secure El-Gamal Encryption I: We apply the Fujisaki-Okamoto transformation-1 [8] to convert the plain El-Gamal encryption scheme $\mathcal{EL}$ into a CCA2 secure scheme which leads to the following scheme:

- $\mathcal{EL}$.`Gen`: The private key and public key pair of a user is $(sk, pk) = (x, g^x)$.
- $\mathcal{EL}$.`Enc`: Compute $r = F(\sigma, m)$, $c_1 = g^r$, $c_2 = \sigma \oplus H(pk^r)$, $c_3 = m \oplus G(\sigma)$ and the ciphertext is $c = \langle c_1, c_2, c_3 \rangle$. H and G are two cryptographic hash functions.
- $\mathcal{EL}$.`Dec`: To decrypt, compute $\sigma = c_2 \oplus H(c_1^{sk})$, $m = c_3 \oplus G(\sigma)$ and accept m if $c_1 \overset{?}{=} g^{F(\sigma,m)}$.

Fact: El-Gamal Encryption I is CCA2 secure if the CDH Problem is hard to solve in polynomial time.

Remark: When the decryption process of this system is attacked by a RAM scraper, the values of c_1^{sk}, σ, m are available to the adversary, assuming that the private key is available in the TPM and the computation of c_1^{sk} (using the private key) is done in the TPM.

3.2 Attack using Glassbox Decryption:

The attack on the specific instance of Fujisaki-Okamoto Transformation-1 follows:

- During the game, the adversary $\mathcal{A}$ chooses two messages m_0 and m_1 and gives them to the challenger $\mathcal{C}$.
- $\mathcal{C}$ chooses a random bit $\delta \in \{0, 1\}$, computes the challenge ciphertext c^* on the message m_δ and sends c^* to $\mathcal{A}$.
- The only restriction posed on $\mathcal{A}$ is that $\mathcal{A}$ should not query the glass box decryption on c^*.
- We show that $\mathcal{A}$, who gets the challenge ciphertext and knows the messages m_0 and m_1 can easily outputs a bit δ' such that $\delta' = \delta$ (δ was selected by $\mathcal{C}$ to generate the challenge ciphertext) by performing the following:
  - Let the challenge ciphertext generated by $\mathcal{C}$ be $c^* = \langle c_1^*, c_2^*, c_3^* \rangle$. Let $c_1^* = g^r$, $c_2^* = \sigma^* \oplus H(pk^r)$ and $c_3^* = m_\delta \oplus G(\sigma^*)$, where $r = F(\sigma^*, m_\delta)$.
  - $\mathcal{A}$ computes $c_1' = c_1^*$, $c_2' = c_2^*$ and chooses c_3' randomly from the range of the hash function $G(.)$ and forms a new ciphertext $c' = \langle c_1', c_2', c_3' \rangle$.
  - Now, $\mathcal{A}$ queries `Glass-Box-Dec`(c').
  - While running the glass box decryption of c', $\mathcal{C}$ generates c_1^{sk}, $\sigma' = \sigma$ and some m' as intermediate values, i.e. $\mathcal{I} = \langle c_1^{sk}, \sigma', m' \rangle$.
  - $\mathcal{C}$ sends $\mathcal{I}$ as response to glass box decryption of c'. Moreover, $\mathcal{C}$ will reject the ciphertext c' because the check $c_1' \overset{?}{=} g^{F(\sigma',m')}$ will fail.
  - Since $\mathcal{A}$ has obtained the value σ' from $\mathcal{C}$, $\mathcal{A}$ can easily obtain the message m_δ by computing $m_\delta = c_3^* \oplus G(\sigma')$ and checking whether $c_1^* \overset{?}{=} g^{F(\sigma',m_\delta)}$.
  - Thus, $\mathcal{A}$ identifies the bit δ almost always.

It should be noted that similar attack can be demonstrated on transformations in [7] and [10].

4 Confronting the RAM Scraper Attack

It can be noted from the above section that almost all the transformations did not withstand RAM scraper attack. It is clear from the description of the attacks that in the decryption process, the private key of the receiver is used to derive a message from the ciphertext and then the validity of the message is checked using some verification step. The attacker who has access to the glass box decryption oracle gets the advantage of the computations done using the private keys of the receiver during the decryption. The motivation behind our scheme is to thwart the attacker from obtaining any useful information due to glass box decryption. One way to do this is to carry out the ciphertext verification before decryption and hence the attacker cannot manipulate the ciphertext in a meaningful way to pass the ciphertext verification test. Thus any manipulated ciphertext gets rejected in the first level of verification itself without any further computations. Even if the ciphertext is tweaked in such a way that the ciphertext verification test passes, the decryption rejects the ciphertext and the intermediate value obtained through the glass box decryption oracle should be designed in such a way that they are of no use to the adversary. Based on this line of thinking, we propose the new scheme that offers security against glass box decryption oracle. We prove the scheme to be secure in the random oracle model.

4.1 The Encryption Scheme ($\texttt{EncryptI}^{\texttt{GB}}$):

The construction of the $\texttt{EncryptI}^{\texttt{GB}}$ scheme makes use of a one-time signature scheme, which is strongly existentially unforgeable under chosen message attack and secure in the random oracle model. The one-time signature scheme is required to be strong because it should have the property that it should not be possible to create a new valid signature even for previously-signed messages. The one-time signature is defined as $\texttt{OSign} = (\texttt{Gen,Sig,Ver})$. Without loss of generality we assume that the $\texttt{Gen}$ algorithm takes the security parameter 1^κ and a random one-time signing key **osk** and produces a value **ovk**, where the one-time signature generating user will use **osk** as the one-time signing key and **ovk** as the corresponding one-time verification key. In other words, $\mathbf{ovk} = \texttt{Gen}(\mathbf{osk}, 1^\kappa)$ can be considered as the key generation algorithm. For every use of **Sig**, as this is a one-time signature algorithm, the signer chooses a fresh **osk**, compute the corresponding **ovk**. Obviously, the function $\texttt{Gen}()$ is a one-way function. $\sigma \leftarrow \texttt{Sig}_{\mathbf{osk}}(m\|\mathbf{ovk})$ is the signing algorithm which outputs the signature σ and $\{0,1\} \leftarrow \texttt{Ver}_{\mathbf{ovk}}(\sigma, m\|\mathbf{ovk})$ is the verification algorithm, which outputs '0' if the signature is invalid and '1' if the signature is a valid. We state informally that any standard signature scheme, which is existentially unforgeable under chosen message attack can be used to construct a strong one-time signature in the random oracle model by just choosing a signing key and a verification key each time to generate a signature and including the verification key along with the message to generate the message digest during the signing process.

The new scheme employs three cryptographic hash functions namely $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_2 : \mathbb{G} \rightarrow \{0,1\}^{\kappa_1+\kappa_2}$ where κ_1 is the length of the message and κ_2 is the length of an element in $\mathbb{Z}_q^*$ and $H_3 : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}$. The details of the construction follows:

- $\texttt{Gen}^{\texttt{GB}}$: The private key and public key pair of a user are generated as follows:
  - Choose $g \in_R \mathbb{G}$ and $x \in_R \mathbb{Z}_q^*$.
  - The private key $\mathbf{sk} = x$ and the public key $\mathbf{pk} = g^x$.
- $\texttt{Enc}^{\texttt{GB}}$: Encryption is done as follows:
  - Choose $r \in_R \mathbb{Z}_q^*$.
  - Compute $\texttt{osk} = H_1(m\|r)$.
  - Compute $c_1 = \texttt{ovk} = g^{\mathbf{osk}}$.
  - Compute $h_3 = H_3(c_1, \mathbf{pk})$, $c_2 = (m\|r) \oplus H_2((h_3\mathbf{pk})^{\mathbf{osk}})$ and $c_3 = h_3^{\mathbf{osk}}$.
  - Compute $\sigma = \texttt{Sig}_{\mathbf{osk}}(c_1, c_2, c_3)$.
  - The ciphertext is $c = \langle c_1, c_2, c_3, \sigma \rangle$.
- $\texttt{Dec}^{\texttt{GB}}$: To decrypt the ciphertext $c = \langle c_1, c_2, c_3, \sigma \rangle$, perform the following:
  - Check whether $\texttt{Ver}_{c_1}(\sigma, \langle c_1, c_2, c_3 \rangle) \stackrel{?}{=} 1$
  - If the above check holds, compute $X = c_1^{\mathbf{sk}} c_3$.
  - Compute $m'\|r' = c_2 \oplus H_2(X)$.
  - Compute $\mathbf{osk}' = H_1(m'\|r')$.
  - Check whether $c_1 \stackrel{?}{=} g^{\mathbf{osk}'}$ and $c_3 \stackrel{?}{=} H_3(c_1, \mathbf{pk})^{\mathbf{osk}'}$, if the check holds then accept m' as the message, else reject the ciphertext c.

Theorem 1. *The encryption scheme $\texttt{EncryptI}^{\texttt{GB}}$ is IND-CCA2 secure with glass box decryption in the random oracle model if the CDH Problem is hard to solve in polynomial time, assuming that the one-time signature scheme $\texttt{OSign}$ is strongly unforgeable.*

Proof: Let the CDH instance given to the challenger $\mathcal{C}$ be $\langle g, g^a, g^b \rangle$. $\mathcal{C}$ gives the description of the system and $\mathbf{pk} = g^a$ as the target public key to the adversary $\mathcal{A}$. $\mathcal{A}$ has to distinguish the ciphertext corresponding to the public key $\mathbf{pk}$.

Phase 1: $\mathcal{A}$ gets access to the following oracles during the first phase of interaction as described below:

$\mathcal{O}_{H_1}$ *Oracle:* Given the input $m\|r$, $\mathcal{C}$ checks whether a tuple of the form $\langle m\|r, \mathbf{osk} \rangle$ exists in list L_{H_1}. If it appears, returns the corresponding $\mathbf{osk}$ else chooses a random value $\mathbf{osk} \in_R \mathbb{Z}_q^*$, stores the tuple $\langle m\|r, \mathbf{osk} \rangle$ in the list L_{H_1} and returns $\mathbf{osk}$.

$\mathcal{O}_{H_2}$ *Oracle:* Given $X \in \mathbb{G}$ as input to the hash oracle, $\mathcal{C}$ checks whether a tuple of the form $\langle X, h_2 \rangle$ exists in list L_{H_2}. If it appears, returns the corresponding h_2 value else chooses a random value $h_2 \in_R \{0,1\}^{\kappa_1+\kappa_2}$, stores the tuple $\langle X, h_2 \rangle$ in the list L_{H_2} and returns h_2.

$\mathcal{O}_{H_3}$ *Oracle:* Given $Y \in \mathbb{G}, \mathbf{pk} \in \mathbb{G}$ as input to the hash oracle, $\mathcal{C}$ checks whether a tuple of the form $\langle Y, \mathbf{pk}, u, h_3 \rangle$ exists in list L_{H_3}. If it appears, returns the corresponding h_3 value else performs the following:

- Chooses a random value $u \in_R \mathbb{Z}_q^*$.
- Computes $h_3 = (\mathbf{pk}^{-1} g^u)$.
- Stores the tuple $\langle Y, \mathbf{pk}, u, h_3 \rangle$ in the list L_{H_3} and returns h_3.

$\mathcal{O}_{\texttt{Glass-Box-Dec}}$ *Oracle:* Given a ciphertext $c = \langle c_1, c_2, c_3, \sigma \rangle$, $\mathcal{C}$ performs the following to decrypt it.

- $\mathcal{C}$ checks whether $\texttt{Ver}_{c_1}(\sigma, \langle c_1, c_2, c_3 \rangle) \overset{?}{=} 1$. If the check doesnot hold, then $\mathcal{C}$ returns all values obtained during the execution of $\texttt{Ver}_{c_1}(\sigma, \langle c_1, c_2, c_3 \rangle)$ to $\mathcal{A}$ and *halts*. (**Note:** The values returned at this step by $\mathcal{C}$ can be obtained by $\mathcal{A}$ even without querying $\mathcal{C}$, because $\texttt{Ver}()$ is a public verification algorithm. If the check holds, $\mathcal{C}$ returns all the values obtained during the execution of $\texttt{Ver}()$ and proceeds.)
- If (there exists a tuple of the form $\langle m'\|r', \mathbf{osk} \rangle$ in the list L_{H_1} with $c_1 = g^{\mathbf{osk}}$) then,
  1. If (there exists a tuple of the form $\langle c_1, \mathbf{pk}, u, h_3 \rangle$ in list L_{H_3}) then,
     - Compute $X = (h_3 \mathbf{pk})^{\mathbf{osk}}$.

     Else,
     - Choose a random value $u \in_R \mathbb{Z}_q^*$.
     - Compute $h_3 = (\mathbf{pk}^{-1} g^u)$.
     - Add the tuple $\langle c_1, \mathbf{pk}, u, h_3 \rangle$ in the list L_{H_3}.
     - Compute $X = (h_3 \mathbf{pk})^{\mathbf{osk}}$.
  2. If (there is no tuple of the form $\langle X, h_2 \rangle$ in list L_{H_2}) then,
     - Choose $h_2 \in_R \{0,1\}^{\kappa_1+\kappa_2}$.
     - Add the tuple $\langle X, h_2 \rangle$ in the list L_{H_2}.
  3. Compute $h_2 \oplus c_3 = \hat{m}\|\hat{r}$
  4. If $(m'\|r' = \hat{m}\|\hat{r})$ then, return $\langle \mathbf{osk}, X, m', r' \rangle$ else return $\langle \mathbf{osk}, X, \hat{m}, \hat{r} \rangle$.

  Else, // (there is no tuple of the form $\langle m'\|r', \mathbf{osk} \rangle$ in the list L_{H_1} with $c_1 = g^{\mathbf{osk}}$)
  1. If (there exists a tuple of the form $\langle c_1, \mathbf{pk}, u, h_3 \rangle$ in list L_{H_3}) then,
     - Compute $X = c_1^u$. // (*Note that even in this case $X = c_1^u$ can be interpreted as $c_1^u = (h_3 \mathbf{pk})^{\mathbf{osk}}$ for some unknown **osk**. However, not knowing **osk** value is not a problem as this was never explicitly set/required in any query. Note again that these kind of computations are internal 'book keeping' computations done by the challenger and adversary need not know any of these details/exact values. Only answers given as a response to the queries should satisfy the equations appearing in encryption steps.*)

     Else,
     - Choose a random value $u \in_R \mathbb{Z}_q^*$.
     - Compute $h_3 = (\mathbf{pk}^{-1} g^u)$.
     - Add the tuple $\langle c_1, \mathbf{pk}, u, h_3 \rangle$ in the list L_{H_3}.

   - Compute $X = c_1^u$.
2. If (there is no tuple of the form $\langle X, h_2 \rangle$ in list L_{H_2}) then,
   - Choose $h_2 \in_R \{0,1\}^{\kappa_1+\kappa_2}$.
   - Add the tuple $\langle X, h_2 \rangle$ in the list L_{H_2}.
3. Compute $h_2 \oplus c_3 = \hat{m} \| \hat{r}$
4. If $(m' \| r' = \hat{m} \| \hat{r})$ then, return $\mathcal{I} = \langle \mathbf{osk}, X, m', r' \rangle$ else return $\mathcal{I} = \langle \mathbf{osk}, X, \hat{m}, \hat{r} \rangle$.

Challenge: $\mathcal{A}$ gives two messages m_0 and m_1 to $\mathcal{C}$. $\mathcal{C}$ chooses $\delta \in \{0,1\}$ and generates the challenge ciphertext as follows:

- Set $c_1^* = g^b$.
- Choose $c_2^* \in_R \{0,1\}^{\kappa_1+\kappa_2}$.
- Choose $u^* \in_R \mathbb{Z}_q^*$, compute $h_3^* = g^{u^*}$, add the tuple $\langle c_1^*, \mathbf{pk}, u^*, h_3^* \rangle$ to the list L_{H_3} and computes $c_3^* = (g^b)^{u^*}$.
- Since the proof is in the random oracle model, $\mathcal{C}$ would be able to generate the signature σ^* without the actual one-time signing key $\mathbf{osk} = b$, with the help of the one-time verification key $\mathbf{ovk} = g^b$. Here, $\mathbf{ovk} = c_1^*$. Thus, we consider that $\mathcal{C}$ is capable of generating the one-time signature $\sigma^* = \mathtt{Sig}_{\mathbf{osk}}(c_1^*, c_2^*, c_3^*)$
- The challenge ciphertext $c^* = \langle c_1^*, c_2^*, c_3^*, \sigma^* \rangle$.

Phase 2: $\mathcal{A}$ makes a second phase of interaction with $\mathcal{C}$ but with the restriction that, $\mathcal{A}$ should not query the decryption of c^*. All oracles are similar to that of Phase 1. Now, $\mathcal{A}$ can submit a ciphertext with few components of the challenge ciphertext c^* intact and altering the other components. Note that if c_1^* is not altered and all other components are altered in the challenge ciphertext c^*, i.e. the altered ciphertext $c' = \langle c_1^*, c_2', c_3', \sigma' \rangle$, then the one-time signature σ' cannot be generated such that it passes the verification as $\mathcal{A}$ does not know the one time signing key $\mathbf{osk}^*$ that corresponds to c_1^*. If $c' = \langle c_1^*, c_2', c_3', \sigma^* \rangle$, this also implies a forgery on $\mathtt{OSig}$ because the message for the one-time signature i.e. $\langle c_1^*, c_2', c_3', \rangle$ is altered but the signature is the same as for the challenge ciphertext c^*. Thus we claim that it is not possible to generate a valid ciphertext with the same one-time verification key c_1^*. Instead, $\mathcal{A}$ can generate a new ciphertext from the challenge ciphertext that passes the verification of the one-time signature alone and this is done as follows:

- Chooses a new one-time signing key $\mathbf{osk}'$ by querying $\mathcal{O}_{H_1}$ with some random input $\overline{m} \| \overline{r}$.
- Computes $c_1' = g^{\mathbf{osk}'}$.
- Queries the $\mathcal{O}_{H_3}$ oracle with $\langle c_1', \mathbf{pk} \rangle$ as input and gets $h_3' = (\mathbf{pk}^{-1} g^{u'})$.
- Computes a new one-time signature as $\sigma' = \mathtt{Sig}_{\mathbf{osk}'}(c_1', c_2^*, c_3^*)$.
- The new ciphertext that passes the one-time signature verification is $c' = \langle c_1', c_2' = c_2^*, c_3' = c_3^*, \sigma' \rangle$.

We show that $\mathcal{A}$ is not going to get any advantage to distinguish the challenge ciphertext c^*, by querying the $\mathcal{O}_{\mathtt{Glass\text{-}Box\text{-}Dec}}$ oracle with c' as input. To decrypt the ciphertext, $\mathcal{C}$ performs the following:

- Checks whether $\mathtt{Ver}_{c_1'}(\sigma', \langle c_1', c_2', c_3' \rangle) \overset{?}{=} 1$. The check holds, and hence $\mathcal{C}$ proceeds with the next step.
- For all tuples of the form $\langle \overline{m} \| \overline{r}, \mathtt{osk}' \rangle$ that appears in list L_{H_1}, checks whether $g^{\mathtt{osk}'} \overset{?}{=} c_1'$. There is obviously a matching tuple because $\mathcal{C}$ has queried the $\mathcal{O}_{H_1}$ oracle with $\overline{m} \| \overline{r}$ as input.
- Checks whether a tuple of the form $\langle Y, \mathbf{pk}, h_3 \rangle$, such that $Y = c_1'$ and $t = c_2'$ appears in list L_{H_3}. Obviously, it appears in the list.
- Computes $X' = (h_3 \mathbf{pk})^{\mathbf{osk}'}$ and checks whether a tuple of the form $\langle X', h_2 \rangle$ exists in list L_{H_2}. There will not be an entry corresponding to this X' in the list L_{H_2} because $\mathcal{A}$ has not queried the $\mathcal{O}_{H_2}$ oracle with X' as input, Hence, $\mathcal{C}$ will reject the ciphertext c'.

– $\mathcal{C}$ returns $\langle X' \rangle$ as the intermediate value. Note that $X' \neq X^*$, where X^* is the value computed during the computation of the challenge ciphertext. $X' = (h_3\mathbf{pk})^{\mathbf{osk}'} = (\mathbf{pk}^{-1}g^{u'}\mathbf{pk})^{\mathbf{osk}'} = g^{u'}$ where as $X^* = (g^{bu^*})\mathbf{pk}^b = (g^{bu^*})g^{ab}$, which clearly shows that $X' \neq X^*$ and X' does not leak any information regarding X^*.

Guess: At the end of the second phase of interaction, $\mathcal{A}$ rejects the ciphertext c^*. Let ϵ be the advantage of $\mathcal{A}$ in breaking the security of our $\texttt{EncryptI}^{\texttt{GB}}$ scheme. $\mathcal{C}$ obtains the solution for the CDH problem with an advantage $\frac{\epsilon}{q_{H_2}}$, where q_{H_2} is the total number of queries made to the $\mathcal{O}_{H_2}$ oracle. This is because:

– $\mathcal{C}$ has set $\mathbf{pk} = g^a$, therefore, the corresponding private key $sk^* = a$ (implicitly). $\mathcal{C}$ has also set $c_1^* = g^b$, therefore, $\mathbf{osk} = b$ implicitly.
– Since $\mathcal{A}$ should have computed $(X = (h_3\mathbf{pk}^*)^{\mathbf{osk}}$ and queried the $\mathcal{O}_{H_2}$ oracle, there should be an entry of the form $\langle X, h_2 \rangle$ in list L_{H_2} such that $X = g^{bu^*}g^{ab}$. $\mathcal{C}$ picks a random X value from the list L_{H_2} and with probability $\frac{1}{q_{H_2}}$ it will be the X value computed by $\mathcal{A}$ to decrypt c^*.
– $\mathcal{C}$ retrieves the value u^* from the list L_{H_3}, which is used to generate the challenge ciphertext and computes $g^{ab} = \frac{X}{(g^b)^{u^*}}$.
– Thus, the probability that $\mathcal{C}$ solves the CDH problem is $\frac{\epsilon}{q_{H_2}}$ ■

4.2 The Encryption Scheme ($\texttt{EncryptII}^{\texttt{GB}}$):

In this section we propose a new public key encryption scheme $\texttt{EncryptII}^{\texttt{GB}}$ and formally prove the IND-CCA2 security with glass box decryption in the standard model. The details of the construction follows:

– $\texttt{Gen}^{\texttt{GB}}$: Let $\mathbb{G}_1$ be a additive group with prime order q and $\mathbb{G}_2$ be a multiplicative group with the same order. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be an admissible bilinear pairing. The scheme uses three cryptographic hash functions defined as: $H_1 : \mathbb{G}_2 \rightarrow \{0,1\}^{l_m}$, $H_2 : \mathbb{G}_1 \times \{0,1\}^{l_m} \rightarrow \mathbb{Z}_q$ and $H_3 : \mathbb{G}_1 \rightarrow \mathbb{Z}_q$. Here l_m is the size of the message. Note that H_3 should be target collision resistant hash function. The private key and public key pair of a user are generated as follows:
  • Choose $x, s \in_R \mathbb{Z}_q$ and $P, Q, Y, Z \in_R \mathbb{G}_1$.
  • Compute $X = xP \in \mathbb{G}_1$.
  • Compute $\alpha = \hat{e}(P,Q)^s \in \mathbb{G}_2$.
  • The private key $\mathbf{sk} = \langle x, s \rangle \in \mathbb{Z}_q^2$ and the public key $\mathbf{pk} = \langle P, Q, X, Y, Z, \alpha \rangle \in \mathbb{G}_1^5 \times \mathbb{G}_2$.
– $\texttt{Enc}^{\texttt{GB}}$: Encryption is done as follows:
  • Choose $r, t \in_R \mathbb{Z}_q$.
  • Compute $C_1 = rP$ and $C_2 = m \oplus H_1(\alpha^r)$.
  • Compute $\hat{h} = H_2(C_1, C_2)$, $h = H_3(r(\hat{h}P + tX))$ and $C_3 = r(hY + Z)$.
  • Set $C_4 = t$.
  • The ciphertext is $C = \langle C_1, C_2, C_3, C_4 \rangle$.
– $\texttt{Dec}^{\texttt{GB}}$: To decrypt the ciphertext $C = \langle C_1, C_2, C_3, C_4 \rangle$, perform the following:
  • Compute $\hat{h} = H_2(C_1, C_2)$.
  • Compute $U = \hat{h}C_1$, $V = C_4 x C_1$ and $h = H_3(U + V)$.
  • If $\hat{e}(C_3, P) \overset{?}{=} \hat{e}(hY + Z, C_1)$ then $m = C_2 \oplus H_1(\hat{e}(C_1, Q)^s)$ else $ABORT$.

Proof of Correctness: To show that the decryption works properly, we have to show that:

1. $U + V = r(\hat{h}P + tX)$.
2. If $C = \langle C_1, C_2, C_3, C_4 \rangle$ is properly constructed, then $\hat{e}(C_3, P) \overset{?}{=} \hat{e}(hY + Z, C_1)$.

3. $\hat{e}(C_1, Q)^s = \alpha^r$, where $C_1 = rP$.

Proof: Assume that for some $r \in \mathbb{Z}_q$,

$$C_1 = rP \tag{1}$$

With respect to the same r,

$$C_3 = r(hY + Z) \tag{2}$$

Hence it should be true that,

$$\hat{e}(C_3, P) \overset{?}{=} \hat{e}(hY + Z, C_1) \tag{3}$$

This proves the second assertion. Now,

$U + V = \hat{h}C_1 + C_4 xC_1 = \hat{h}rP + txrP = r(\hat{h}P + txP) = r(\hat{h}P + tX)$

Thus,

$$U + V = r(\hat{h}P + tX) \tag{4}$$

This shows that $h = H_3(U+V)$ correctly recovers the h computed in the encryption algorithm. This proves the first claim. For the third claim, we note that $\hat{e}(C_1, Q)^s = \hat{e}(rP, Q)^s = [\hat{e}(P, Q)^s]^r = \alpha^r$, Therefore,

$$\hat{e}(C_1, Q)^s = \alpha^r \tag{5}$$

This completes the proof that the decryption correctly recovers the message.

Remark: The glass box decryption oracle should return $\langle \hat{h}, U, V, h, hY + Z, \hat{e}(C_1, Q)^s \rangle$. In fact, $\hat{h}$ and U can be computed by the addversary himself. Note that, the computation of V and $\hat{e}(C_1, Q)^s$ involves one way function evaluation of private key values. Also, $\hat{e}(C_1, Q)^s$ need to be sent only if the test in the last step of decryption algorithm passes.

Theorem 2. *The encryption scheme* $\texttt{EncryptII}^{\texttt{GB}}$ *is IND-CCA2 secure with glass box decryption, if the DBDH Problem is hard to solve in polynomial time.*

Proof: Let $\langle (R, aR, bR, cR) \in \mathbb{G}_1^4, \gamma \in \mathbb{G}_2 \rangle$ be an instance of the DBDH problem. Let $\mathcal{A}$ be an algorithm that is capable of breaking the security of the $\texttt{EncryptII}^{\texttt{GB}}$ scheme. We will show how to construct another algorithm that uses $\mathcal{A}$ to break the DBDH problem in $\mathbb{G}_1$ and $\mathbb{G}_2$. This algorithm is called the challenger $\mathcal{C}$. $\mathcal{A}$ interacts with $\mathcal{C}$ and $\mathcal{C}$ responds to $\mathcal{A}$'s queries as given below:

Setup: $\mathcal{C}$ sets up a system as follows:

– Set $P = R$
– Set

$$Q = bR \tag{6}$$

– Set

$$\alpha = \hat{e}(aR, bR) \tag{7}$$

Therefore, $\alpha = \hat{e}(aR, bR) = \hat{e}(R, bR)^a = \hat{e}(P, Q)^a$

Thus, the second component of the private key denoted as s, is in fact a (implicitly). $\mathcal{C}$ does not know the value of a. Now, choose $x \in_R \mathbb{Z}_q$ and set

$$X = xP \tag{8}$$

This fixes the first component of the private key. Thus the private keys are $\langle x, s = a \rangle$ and $\mathcal{C}$ knows x but does not know s.

$\mathcal{C}$ chooses $\tilde{h}, y, \tilde{z} \in_R \mathbb{Z}_q$ and computes

$$\beta = \tilde{h}(cP) \tag{9}$$

$$h^* = H_3(\beta) \tag{10}$$

$$Y = \frac{1}{h^*}(Q + yP) \tag{11}$$

$$Z = -Q + \tilde{z}P \tag{12}$$

The public keys are $\langle P, Q, X, Y, Z, \alpha \rangle$ and the private keys are $\langle x, s = a \rangle$

Phase I: $\mathcal{A}$ performs a series of queries to the glass box decryption oracle. The description of the glass box decryption oracle and the responses given by $\mathcal{C}$ to the queries by $\mathcal{A}$ are described below.

$\mathcal{O}_{\texttt{Glass-Box-Dec}}$ *Oracle:* When a request for decryption of ciphertext $C = \langle C_1, C_2, C_3, s \rangle$ is made, $\mathcal{C}$ decrypts it in the following way:

– Computes

$$\hat{h} = H_2(C_1, C_2) \tag{13}$$

$$U = \hat{h}C_1 \tag{14}$$

Since, $\mathcal{C}$ knows the private key x, $\mathcal{C}$ can also compute

$$V = C_4 x C_1 \tag{15}$$

Since the values of U and V are correct, $\mathcal{C}$ computes correctly

$$h = H_3(U + V) \tag{16}$$

Note that H_3 is a target collision resistant hash function and if $(h = h^*)$, abort. Since the Y and Z values are public $\mathcal{C}$ computes correctly the value

$$hY + Z \tag{17}$$

So far, $\mathcal{C}$ could do all computations directly as all the input values are correctly available. So, there is no difficulty in computing and returning to $\mathcal{A}$ the values $\langle \hat{h}, U, V, h, hY + Z \rangle$. However, if the check $\hat{e}(C_3, P) \overset{?}{=} \hat{e}(hY + Z, C_1)$ passes, $\mathcal{C}$ must return the value $\hat{e}(C_1, Q)^s$ as well to $\mathcal{A}$, However, as noted before $\mathcal{C}$ does not know the value of s. Thus, $\mathcal{C}$ has to simulate this value interms of other quantities computable by $\mathcal{C}$. Observe that since P is a generator,

$$C_1 = rP,\ for\, some\, r \in \mathbb{Z}_q \tag{18}$$

Since $\hat{e}(C_3, P) = \hat{e}(hY + Z, C_1)$ it follows that

$$C_3 = r(hY + Z) \tag{19}$$

For the same r defined in equation (18). Now,

$$\begin{aligned}\hat{e}(C_1, Q)^s &= \hat{e}(rP, Q)^s \\ &= \hat{e}(P, Q)^{rs} \\ &= \hat{e}(sP, Q)^r \\ &= \hat{e}(aP, rQ)^s \text{ Since } (s = a)\end{aligned}$$

$\mathcal{C}$ knows the value of $aP = aR$ and value of Q as they are inputs for the hard problem. However, $\mathcal{C}$ does not know the value of r. Hence $\mathcal{C}$ will compute the value of rQ indirectly in terms of other values known to $\mathcal{C}$. From equations (11), (12) and (19),

$$\begin{aligned}
C_3 &= r(hY + Z) \\
&= r\left(\frac{h}{h^*}(Q + yP) - Q + \tilde{z}P\right) \\
&= \left(\frac{h}{h^*} - 1\right) rQ + \left(\frac{h}{h^*}y + \tilde{z}\right) rP \\
&= \left(\frac{h}{h^*} - 1\right) rQ + \left(\frac{h}{h^*}y + \tilde{z}\right) rP
\end{aligned}$$

Rearranging, we obtain

$$rQ = \left(\frac{h}{h^*} - 1\right)^{-1} \left[C_3 - \left(\frac{h}{h^*}y + \tilde{z}\right) C_1\right] \tag{20}$$

Observe that all values in the RHS of equation (20) is available to $\mathcal{C}$. Hence rQ can be computed using equation (20). Hence rQ can be computed using equation (20). Thus, $\hat{e}(C_1, Q)^s = \hat{e}(aP, rQ)$ can be computed even without knowing s. Hence, the glass box decryption queries can be perfectly answered by $\mathcal{C}$. That is $\mathcal{C}$ can give perfect training to $\mathcal{A}$.

Challenge: On getting sufficient training from $\mathcal{C}$ in ***Phase I*** interaction, $\mathcal{A}$ gives $\mathcal{C}$ two messages m_0, m_1 of equal length. $\mathcal{C}$ computes the ciphertext C^* by performing the following steps:

– Set
$$C_1^* = cR = cP \tag{21}$$
cR is the input to the hard problem.

– Compute
$$C_2^* = m_\delta \oplus H_1(\gamma) \tag{22}$$
Here, $\delta \in \{0, 1\}$ is a random bit and γ is an input to the hard problem

– Compute
$$C_3^* = yC_1^* + \tilde{z}C_1^* \tag{23}$$

– Compute
$$C_4^* = (\hat{h} - \tilde{h})x^{-1} \tag{24}$$
Where, $\hat{h} = H_2(C_1^*, C_2^*)$ and $\tilde{h}$ was chosen by $\mathcal{C}$ at setup time. x is one of the private keys known to $\mathcal{C}$.

– The challenge ciphertext $C^* = \langle C_1^*, C_2^*, C_3^*, C_4^* \rangle$ is send to $\mathcal{A}$.

Lemma 1. *The challenge ciphertext $C^* = \langle C_1^*, C_2^*, C_3^*, C_4^* \rangle$ is a valid and properly formed ciphertext.*

Proof: Since $C_1^* = cP$, we should show that

$$C_3^* = c(hY + Z) \tag{25}$$

Where, $h = H_3(c(\hat{h}P + tX))$ and $C_4^* = t = (\hat{h} - \tilde{h})x^{-1}$ Now,

$$\begin{aligned}
c(\hat{h}P + tX) &= c(\hat{h}P + C_4^*X) \\
&= c(\hat{h}P + (\hat{h} - \tilde{h})x^{-1}xP) \text{ (From equation (24)} \\
&= c(\hat{h}P + \hat{h}P - \tilde{h}P) \\
&= \tilde{h}(cP) = \beta \text{ (From equation (9))}
\end{aligned}$$

Therefore,

$$h = H_3(c(\hat{h}P + tX)) = H_3(\beta) = h^* \tag{26}$$

From equations (23) and (26) we conclude that $C^* = \langle C_1^*, C_2^*, C_3^*, C_4^* \rangle$ as defined above will be a valid/consistant ciphertext, if we show $C_3^* = c(h^*Y + Z)$. C_3^* was computed as $yC_1^* + \tilde{z}C_1^*$ in equation (23). Thus we have to show that:

$$c(h^*Y + Z) = yC_1^* + \tilde{z}C_1^* \tag{27}$$

In fact,

$$\begin{aligned} c(h^*Y + Z) &= c[Q + yP - Q + \tilde{z}P] \text{ (From equations (11) and (12))} \\ &= y(cP) + \tilde{z}(cP) \\ &= yC_1^* + \tilde{z}C_1^* \end{aligned}$$

This completes the proof that $C^* = \langle C_1^*, C_2^*, C_3^*, C_4^* \rangle$ is a valid/consistant ciphertext. □

Phase II: After receiving the challenge ciphertext C^*, $\mathcal{A}$ is allowed to perform further glass box decryption oracle queries, with the constraint that, $\mathcal{A}$ should not query the decryption of C^*.

Guess: Recall that the instance of the hard problem $\langle R, aR, bR, cR, \gamma \rangle$ are used by $\mathcal{C}$ as $P = R$, $Q = bR$ and $\alpha = e(a\hat{R}, bR) = \hat{e}(P, Q)^s$ while setting the system. When constructing the challenge ciphertext $C_1^* = cR = rP$ and $C_2^* = m_{\delta \oplus H_2(\gamma)}$. If m_δ were correctly identified by $\mathcal{A}$, the implicitly, by the collision resistant property of H_2, $\gamma = \alpha^r = \alpha^c = \hat{e}(P, Q)^{ac} = \hat{e}(R, bR)^{ac} = \hat{e}(R, R)^{abc}$

Thus, $\mathcal{C}$ obtains the solution to the DBDH problem instance with almost the same advantage of $\mathcal{A}$. ■

5 Conclusion

The security industry is constantly facing newer challenges due to advancement in technology and in the understanding of the software systems. Side channel attacks, key leakages [12] etc are among new kinds of threats and these mechanisms provide adversary many information that are not captured by the CCA2 attack model. Thus, looking beyond the CCA2 security in a formal way to counter newer kind of threats is an important area of research. RAM scrapers allows one to obtain many values other than the output when we execute an algorithm. Specifically, when a decryption algorithm is run, RAM scrapers may expose to adversary several values depending on the secret key and these values may enhance the breaking powers of adversary significantly. We refer the process of obtaining intermediate values in addition to the output as glass box execution and RAM scrappers have made the glass box execution a reality. We have initiated the study of security under glass box decryption. We have shown that if glass box decryption is available, we can break even the CCA2 secure systems, indicating the fact that IND-CCA2 security with glass box decryption attack is strictly stronger than IND-CCA2 attack. We have showed that one of the popular transformations that successfully construct a CCA2 secure system, fails to provide security against powerful RAM scrapers. Finally, we have proposed two schemes and proved them to be secure against this attack under the random oracle model and standard model respectively.

References

1. Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *Theory of Cryptography, TCC 2009*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009.
2. Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 36–54. Springer, 2009.

3. Wade Baker at al. 2010 data breach investigations report, http://www.verizonbusiness.com/resources/reports/rp_2010-data-breach-report_en_xg.pdf, 2010.
4. Dan Boneh and Jonathan Katz. Improved efficiency for cca-secure cryptosystems built using identity-based encryption. In *Topics in Cryptology - CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 87–103. Springer, 2005.
5. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2004.
6. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.
7. Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In *Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68. Springer, 1999.
8. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.
9. Ali Juma and Yevgeniy Vahlis. Protecting cryptographic keys against continual leakage. In *Advances in Cryptology - CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 41–58. Springer, 2010.
10. Eike Kiltz and John Malone-Lee. A general construction of ind-cca2 secure public key encryption. In *Cryptography and Coding, 9th IMA International Conference*, volume 2898 of *Lecture Notes in Computer Science*, pages 152–166. Springer, 2003.
11. Eike Kiltz and Krzysztof Pietrzak. Leakage resilient elgamal encryption. In *Advances in Cryptology - ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 595–612. Springer, 2010.
12. Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 18–35. Springer, 2009.
13. Ravi Sethi. *Programming Languages - Concepts and Constructs (Second Edition)*. Addison-Wesley Publishing, 1996.