SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2016; **9:**1650–1662
Published online 17 April 2016 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1454

SPECIAL ISSUE PAPER

Stronger public key encryption system withstanding RAM scraper like attacks

Sree Vivek Sivanandam[1*†], Sharmila Deva Selvi Selvaraj[2†], Akshayaram Srinivasan[3†] and Pandu Rangan Chandrasekaran[4]

[1] Samsung R&D Institute, Bangalore, India.
[2] Microsoft Research, Bangalore, India.
[3] University of California, Berkeley, CA, U.S.A.
[4] Indian Institute of Technology Madras, Chennai, India

ABSTRACT

The indistinguishability of ciphertext under the chosen ciphertext attack (IND-CCA2) is often considered to offer the strongest security notion for a public key encryption system. Nowadays, because of the availability of powerful malwares, an adversary is able to obtain "more" information than what he could obtain in the CCA2 security model. In order to realistically model the threats posed by such malwares, we need to empower the adversary to obtain additional information. This paper initiates a research to counter malwares such as RAM scrapers and extend the CCA2 model with oracles providing additional information to capture the effect of RAM scrapers precisely. We call this more stronger security notion as *glass box decryption*. After discussing the new kind of attack/threat and the related oracle, we show that almost all CCA2 secure systems are vulnerable to this kind of attack. We then propose a new system that offers security against glass box decryption and provide the formal security proof for the new system in the standard model. Copyright © 2016 John Wiley & Sons, Ltd.

KEYWORDS
Public Key Encryption; IND-CCA2; Glass Box Decryption; Standard Model; RAM Scraper

***Correspondence**
Sree Vivek Sivanandam , Samsung R&D Institute, Bangalore, India.
E-mail: ssreevivek@gmail.com

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 to an adversary who carries an adaptive chosen ciphertext attack (IND-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 decryption oracle. However, in real life, an adversary could obtain additional related information like some bits of the secret key or the values of ephemeral keys through other sources/channels. This additional information might help the adversary to completely break the security system. The side channel attacks, for example, enables an adversary to break the system if he is in physical proximity of the device performing decryption/key generation. Although cryptosystems may be proved secure in the "traditional" sense, leakage of such information to the adversary would render them completely insecure. Thus, it is important to extend the existing models to capture this additional information leakage to accurately model a real-world adversary.

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.

1.1. RAM scraper

RAM scraper is a malware created to grab data residing in a system's 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

† Part of work done while the author was a student at IIT-Madras.

threats posed by RAM scrapers were added to the list of *Top Data Breach Attacks* by Verizon Business [1].

Because all the standards for secure communication of sensitive data require end-to-end encryption while being transmitted, received, or stored, the unencrypted information or the ephemeral values residing in RAM during decryption provide an easy target for the adversary. For this specific reason, RAM scrapers could be used with a devastating effect by adversaries against encryption systems.

1.2. Hybrid computing environment

A traditional computing system typically consists of a programmable general purpose computing device. But nowadays, there are several hardware products (called as trusted hardware machines or secure co-processors) that are particularly designed (like IBM crypto cards) to provide enhanced security for cryptographic applications. These trusted machines usually come with a small memory and limited computing power. Therefore, it is highly impractical to perform the entire computation on these trusted hardware products.

We consider a "hybrid" computation model where the computation is divided between a trusted machine and a normal computing machine. The secret keys are stored in the trusted machine. Figure 1 shows such a computing model. Cryptographic algorithms, specifically the decryption algorithm, can be broken into two parts. The part involving computations with the secret key can be executed in Trusted Platform Module (TPM), while the other part (that does not use secret keys directly) can be carried out in a less trusted environment that is outside the TPM. This approach is often used to increase the throughput and lower the latency while still protecting the secret keys inside TPM. This approach is similar to the *Partitioned Computational Model* with secure co-processor [2]. Ever since the TPM has become a practical reality, breaking a functionality between trusted and untrusted platforms in a hybrid environment has become an active area of research. Some of the recent results [3] have implemented a hybrid computation model in practice.

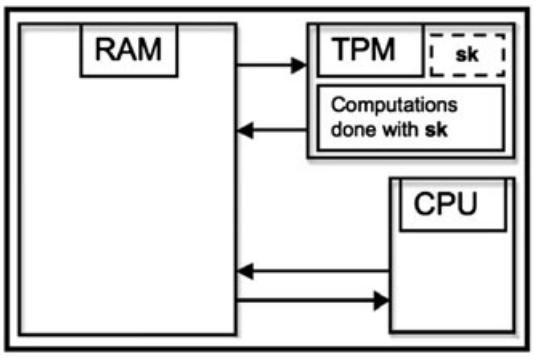

Figure 1. A hybrid system with trusted platform module deployed.

1.3. Glass box decryption

The key question that we consider in this work is

How to model RAM scraper like-attacks in the hybrid computation model?

In order to address the aforementioned question, in the context of public key encryption, 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 computed values available outside the TPM during the execution of the decryption algorithm. We make the realistic assumption that the adversary does not obtain the private key directly through any glass box decryption oracle query.

In a hybrid environment, we need to specify explicitly which part of the system executes a particular step of a program. Specifically, we label the steps executed in the TPM by secure world computation (SC) and all the steps done outside TPM by normal world computation (NC). $\texttt{Glass-Box-Dec}_{sk}(c)$ would return all the values generated by NC during the execution of the decryption algorithm on c, besides the values that are sent for NC by TPM. If the decryption algorithm is aborted, whatever values generated/available in NC until the abortion will be returned.

Remark 1. In a public key encryption system, the sender (or the encrypter) may be physically located far off from the intended receiver, and the sender's credentials are completely hidden. This is precisely the reason that we do not consider a glass box oracle for the encryption algorithm as it is unrealistic to consider a real-world scenario where the attacker has access to the RAM of the sender.

Remark 2. We assume that the entire decryption algorithm is not executed in TPM and some part of the computation would be done outside. Note that if the entire decryption algorithm is executed completely in TPM, then the effect of glass box decryption oracle is identical to black box decryption oracle. Hence, the values returned by glass box decryption oracle depends on the implementation and, specifically, on the way the hybrid system partitions and carries out the execution of the decryption algorithm.

1.4. Related work

Leakage-resilient cryptography [4] is a well-studied cryptographic primitive, which was introduced to model adversaries who might be able to obtain some non-black box information due to close physical presence to a device performing decryption (side-channel attacks). In such models, the adversary is allowed to obtain a bounded number of bits of the secret key in an adaptive way. The threat model, which we consider in this work, differs from that of leakage resilience in two important ways. Firstly, side-channel attacks require close physical proximity to the computing device in some form of sensors measuring the computation

time or other parameters like energy dissipated. On the other hand, RAM scraper can be deployed remotely, and the adversary would continuously be getting feedback on the contents of the RAM memory. Secondly, we consider a situation where all the bits of the secret key are completely hidden from the adversary and only the temporary values computed outside the TPM during decryption are made available. Leakage-resilient models consider a scenario where some bits of the secret key are made available but no other information regarding the ephemeral values are given out. Given the aforementioned differences, leakage-resilient models do not seem to exactly capture the threat due to malwares such as RAM scraper, and hence, we propose a new model to capture such scenarios. It would be an interesting future work to consider situations that combine both leakage resilience as well as glass box decryption where the adversary obtains some bits of the secret key as well as some of the intermediat computation values.

In order to counter the adversary who has access to the full decryption algorithm, researchers have explored the idea of obfuscation in the past few years. Obfuscation of a program involves creating a functionally equivalent but "unintelligible" program so that observing the execution of the program gives the adversary no information about the internal states. The notion of program obfuscation was formalized in the seminal work of Barak *et al*. in 2001 [5]. They also showed a negative result that the strongest notion of obfuscation considered by them is impossible to achieve for arbitrary functionalities. However, candidate construction of obfuscation systems satisfying milder security notions have been proposed in the seminal work by Garg *et al*. in [6]. Obfuscations are, in general, extremely difficult to achieve even for extremely trivial functions. Several weaker variations and heuristics are proposed in the literature, but none of them are found to be practical. An extensive experimental study reported in [7] clearly exposes the impracticality of obfuscation. Thus, we completely rule out the possibilities of using obfuscation to counter the RAM scrapers like adversaries. As noted earlier, the models for side-channel attacks and leakage models are not appropriate for studying RAM scraper like malware because those models are defined for exposure of some bounded amount of bits of secret keys, whereas RAM scrapers obtain all bits (complete values) of the values in RAM and obtains *no bits of the secret keys*.

1.5. Our contribution

Our contributions can be summarized as follows:

1.5.1. Glass box decryption.

We extend the security notion of IND-CCA2 to capture RAM scraper-like attacks. In particular, we replace the *black box* decryption oracle provided in the CCA2 definition with a *glass box* decryption oracle. The glass box decryption oracle query on a ciphertext, 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 values available/computed outside the TPM during the execution of the decryption algorithm will be returned.
- If the decryption algorithm is terminated with an *ABORT*, then the values available/computed outside the TPM up to that point will be returned.
- The private key values will not be returned.

1.5.2. Vulnerability with existing adaptive chosen ciphertext Secure systems.

We show that a "natural" implementation (in the hybrid computing system) of the popular method of converting a chosen plaintext attack secure encryption to a CCA secure encryption proposed by Fujisaki and Okamoto in [8] when applied to the ElGamal encryption system is insecure when given access to the glass box decryption oracle. We note that even if "most" of the computations are performed in the TPM, Fujisaki–Okamoto transformation applied on the ElGamal system is insecure in the glass box decryption model. Further, we show that "natural" implementations of Cramer–Shoup [9] and Klitz–Malone Lee transformation are insecure.

1.5.3. A glass box secure system.

We then propose a new encryption scheme, which is adaptive chosen ciphertext secure even in the presence of glass box decryption oracle access. We prove the security of the system in the standard model under the decisional bilinear Diffie–Hellman (DBDH) assumption. We show the security of our system even for an implementation that carries out minimal computation in TPM and bulk of remaining computations done in insecure normal world that is outside TPM.

2. PRELIMINARIES

A function $\mu(\cdot) : \mathbb{N} \rightarrow \mathbb{R}^+$ is said to be negligible if for every positive polynomial $p(\cdot)$, there exists an N such that for all $n \geq N$, $\mu(n) < 1/p(n)$. If X is a finite set, we denote $x \in_R X$ the process of sampling an element x uniformly at random from X. PPT machines refer to Probabilistic Polynomial Time Turing machines. All PPT machines run in time polynomial in the security parameter denoted by κ.

We first define the notion of computational indistinguishability of two distribution ensembles.

Definition 1. *$\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are computationally indistinguishable, denoted by $\{X_n\}_n \overset{c}{\approx} \{Y_n\}_n$, if for all PPT machines (distinguishers) D, there exists a negligible function $\mu(\cdot)$ such that for all $n \in \mathbb{N}$,*

$$\left|Pr[b \leftarrow D(X_n) : b = 1] - Pr[b \leftarrow D(Y_n) : b = 1]\right| \leq \mu(n)$$

The quantity $\left|Pr[b \leftarrow D(X_n) : b = 1] - Pr[b \leftarrow D(Y_n) : b = 1]\right|$ is termed as the *advantage* the distinguisher D has in distinguishing X_n from Y_n.

We now recall the definition of pseudorandom generator from [10].

Definition 2. *A pseudorandom generator G is a deterministic polynomial time algorithm satisfying the following two conditions:*

$$\left\{ \begin{array}{l} (p, \mathbb{G}_1, \mathbb{G}_2, R, e) \leftarrow \mathsf{Gen}(1^{\kappa}); \\ a, b, c \overset{\$}{\leftarrow} \mathbb{Z}_p^* : \\ (R, aR, bR, cR, e(R,R)^{abc}) \end{array} \right\}_{\kappa} \overset{c}{\approx} \left\{ \begin{array}{l} (p, \mathbb{G}_1, \mathbb{G}_2, R, e) \leftarrow \mathsf{Gen}(1^{\kappa}); \\ a, b, c, d \overset{\$}{\leftarrow} \mathbb{Z}_p^* : \\ (R, aR, bR, cR, e(R,R)^{d}) \end{array} \right\}_{\kappa}$$

(1) ***Expansion:*** *There exists a function* $l : \mathbb{N} \to \mathbb{N}$ *such that* $l(\lambda) > \lambda$ *for all* $\lambda \in \mathbb{N}$ *and* $|G(s)| = l(|s|)$ *for all* $s \in \{0,1\}^*$.

(2) ***Pseudorandomness:*** $\{G(U_\lambda)\}_\lambda \overset{c}{\approx} \{U_{l(\lambda)}\}_\lambda$ *where* U_n *denotes the uniform distribution on* $\{0,1\}^n$.

The function l is called as the expansion factor.

2.1. Decisional bilinear Diffie–Hellman problem

Let Gen be an algorithm that takes 1^{κ} as input and randomly generates the parameters $(p, \mathbb{G}_1, \mathbb{G}_2, R, e)$ where p is a κ bit prime, $\mathbb{G}_1$ is a additive group of order p , R is a generator for $\mathbb{G}_1$, $\mathbb{G}_2$ is a multiplicative group of order p, and e is a bi-linear map from $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.

Definition 3. *The DBDH assumption states that the following distribution ensembles are computationally indistinguishable:*

2.2. Indistinguishable encryptions under adaptive chosen ciphertext attack security notion

Definition 4. *Let* $\mathcal{E} = (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ *be a public key encryption system. Let us define the following experiment.*

We say that $\mathcal{E}$ *is IND-CCA2 if for all probabilistic polynomial time adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *where* $\mathcal{A}_2$ *does not query* c^* *to* $\mathtt{Dec}_{sk}(.)$

$$.|Pr[IND - CCA2(\mathcal{E}, \mathcal{A}, \kappa) = 1] - 1/2| \leq negl(\kappa)$$

$$IND - CCA2(\mathcal{E}, \mathcal{A}, \kappa)$$

- $(sk, pk) \leftarrow \mathtt{KeyGen}(1^{\kappa})$.
- $(m_0, m_1, state) \leftarrow \mathcal{A}_1^{\mathtt{Dec}_{sk}(.)}(pk)$
- $\delta \in_R \{0,1\}$
- $c^* \leftarrow \mathtt{Enc}_{pk}(m_\delta)$
- $\delta' \leftarrow \mathcal{A}_2^{\mathtt{Dec}_{sk}(.)}(state, c^*)$.
- Output 1 if $\delta = \delta'$ and 0 otherwise.

$$IND - CCA2^{GB}(\mathcal{E}, \mathcal{A}, \kappa)$$

- $(sk, pk) \leftarrow \mathtt{KeyGen}(1^{\kappa})$.
- $(m_0, m_1, state) \leftarrow \mathcal{A}_1^{\mathtt{Glass\text{-}Box\text{-}Dec}_{sk}(.)}(pk)$
- $\delta \in_R \{0,1\}$
- $c^* \leftarrow \mathtt{Enc}_{pk}(m_\delta)$
- $\delta' \leftarrow \mathcal{A}_2^{\mathtt{Glass\text{-}Box\text{-}Dec}_{sk}(.)}(state, c^*)$.
- Output 1 if $\delta = \delta'$ and 0 otherwise.

2.3. Indistinguishable encryptions under adaptive chosen ciphertext attack 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 $\mathcal{I} \leftarrow$ Glass-Box-$\mathtt{Dec}_{sk}(c)$ denotes all the values computed/used in RAM (outside TPM) until *ABORT*/termination of the execution of decryption algorithm on the ciphertext c. If the execution does not *ABORT*, then the decrypted message is also included in $\mathcal{I}$.

Definition 5. *Let $\mathcal{E}$ = (KeyGen, Enc, Dec) be a public key encryption system. Let us define the following experiment.*
We say that $\mathcal{E}$ has IND-CCA2 in the glass box model if for all probabilistic polynomial time adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ where $\mathcal{A}_2$ does not query c^* to Glass-Box-$\mathtt{Dec}_{sk}(.)$,:

$$2.|Pr[IND - CCA2^{G}B(\mathcal{E}, \mathcal{A}, \kappa) = 1] - 1/2| \leq negl(\kappa)$$

3. VULNERABILITY IN AN IMPLEMENTATION OF AN ADAPTIVE CHOSEN CIPHERTEXT ATTACK SECURE SCHEME

The ElGamal encryption scheme [11] over a cyclic group $\mathbb{G}$ of prime order p works as follows. The public key consists of a generator $g \in_R \mathbb{G}$ and $h = g^x$, where $x \in_R \mathbb{Z}_p$ is the secret key sk. The public key $pk = \langle g, h\rangle$. Encryption defines the ciphertext as $c=\langle c_1, c_2\rangle=\langle g^r, h^r \oplus m\rangle$, where $r \in_R \mathbb{Z}_p$. Decryption reconstructs the message by computing $m = c_2 \oplus c_1^{sk}$. The following is a specification of CCA2 secure system obtained after transforming the basic chosen plaintext attack secure ElGamal encryption scheme by Fujisaki–Okamoto transformation [8]:

- $\mathcal{EL}$.Gen: The private key and public key pair of a user is $(sk, pk) = (x, g^x)$.
- $\mathcal{EL}$.Enc: Choose σ randomly from $\{0,1\}^{l_m}$. 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 $c = \langle c_1, c_2, c_3\rangle$. F, H, G are 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 $c_1 \stackrel{?}{=} g^{F(\sigma,m)}$.

The aforementioned specification of decryption assumes that the decryption algorithm is executed in a single execution module (without a TPM). We now specify an implementation of the decryption algorithm in a system that has TPM. We assume that TPM is a very compact module with resource constraints and that we get the minimal computations done in TPM. As mentioned earlier, all steps computed inside TPM will be labeled by SC and all other steps will be labeled by NC. The glass box decryption oracle will return all values computed in steps labeled NC.

The description of the decryption algorithm follows: $\mathcal{EL}$.Dec: TPM→RAM represents the data transfer from TPM to RAM and RAM→TPM represents the data transfer from RAM to TPM. To decrypt the ciphertext:

- SC: Compute $temp_1 = c_1^{sk}$
- TPM→RAM: $temp_1$
- NC: Compute $H(temp_1)$
- NC: Compute $\sigma = c_2 \oplus H(temp_1)$
- NC: Compute $G(\sigma)$
- NC: Compute $m = c_3 \oplus G(\sigma)$
- NC: Compute $F(\sigma, m)$
- NC: Accept m if $c_1 \stackrel{?}{=} g^{F(\sigma,m)}$

Remark 3. When the glass box decryption oracle is queried on a ciphertext c, the value in the computation labeled NC, namely, $\mathcal{I} = \langle temp_1, H(temp_1), \sigma, G(\sigma), m, F(\sigma, m)\rangle$ is returned as response.

Attack using glass box decryption: The following attack can be mounted on the above implementation of Fujisaki–Okamoto transformation.

- The adversary $\mathcal{A}_1$ chooses two messages m_0 and m_1.
- A random bit $\delta \in \{0, 1\}$ is chosen, and the challenge ciphertext c^* on the message m_δ is given to $\mathcal{A}_2$.
- Now, $\mathcal{A}_2$ finds δ as follows:
  - Let the challenge ciphertext generated be $c^* = \langle c_1^*, c_2^*, c_3^*\rangle$, where $c_1^* = g^r$, $c_2^* = \sigma^* \oplus H(pk^r)$ and $c_3^* = m_\delta \oplus G(\sigma^*)$, where $r = F(\sigma^*, m_\delta)$.
  - $\mathcal{A}_2$ constructs $c_1' = c_1^*$, $c_2' = c_2^*$ 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, $\mathcal{A}_2$ queries Glass-Box-Dec(c').
  - While running the glass box decryption of c', $\mathcal{I} = \langle temp_1, H(temp_1), \sigma', G(\sigma'), m', F(\sigma', m')\rangle$ is generated for some m'.
  - $\mathcal{I}$ is sent out as response to glass box decryption of c'. (Note that the oracle may accept or reject m', which is immaterial for the attack).
  - Since $c_1' = c_1^*$ and $c_2' = c_2^*$ it is clear that $\sigma' = \sigma^*$. Hence, $\mathcal{A}_2$ can recover m_δ by computing $m_\delta = c_3^* \oplus G(\sigma')$
  - Thus, $\mathcal{A}_2$ identifies the bit δ always.

Remark 4. It may appear as though the attack is possible because a lot of computations are done outside TPM. However, it is easy to see that even if the only step performed outside the TPM is the last step, this would enable $\mathcal{A}$ to break the system. The appendix has similar case studies on implementation of the Cramer–Shoup cryptosystem

and another transformation reported in [12] under minimal computation done outside TPM. It is also easy to realize this attack on an implementation of the transformation in [13]. Thus, we arrive at an important conclusion that the entire decryption algorithm must be executed inside TPM to safeguard against RAM scraper-like attacks. However, this may be infeasible if TPM has memory and computational constraints.

In Appendix A and B, we show that "natural" implementations of the Cramer–Shoup system and Klitz–Malone Lee transformations are not glass box secure, respectively.

4. CONFRONTING THE RAM SCRAPER ATTACK

It can be noted from the previous section that "natural" implementations of well known CCA2 systems 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 during 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 values computed outside the TPM 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 the aforementioned intuition, we propose a new system that offers security against glass box decryption oracle. We prove the security of the system under the decisional bilinear Diffie–Hellman assumption.

4.1. The encryption scheme ($\texttt{EncryptI}^{\texttt{GB}}$):

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

- $\texttt{Setup}^{\texttt{GB}}$: Let $(p, \mathbb{G}_1, \mathbb{G}_2, P, e) \leftarrow \mathsf{Gen}(1^\kappa)$. The scheme uses a pseudorandom generator $H_1 : \mathbb{G}_2 \rightarrow \{0,1\}^{l_m}$ and two cryptographic hash functions defined as: $H_2 : \mathbb{G}_1 \times \{0,1\}^{l_m} \rightarrow \mathbb{Z}_q$ and $H_3 : \mathbb{G}_1 \rightarrow \mathbb{Z}_q$. H_3 is additionally target collision resistant hash. Here $l_m > \log q$ is the size of the message.
- $\texttt{KeyGen}^{\texttt{GB}}$: The private key and public key pair of a user are generated as follows:
- $\texttt{Enc}^{\texttt{GB}}$: Encryption is done as follows:
- $\texttt{Dec}^{\texttt{GB}}$: We describe the implementation of the decryption algorithm in conventional system and hybrid system. To decrypt the ciphertext $C = \langle C_1, C_2, C_3, C_4 \rangle$, perform the following:

Remark 5. A glass box decryption oracle would expose all the values computed and used in the NC, namely, $\mathcal{I} = \langle \hat{h}, U, V, h, e(C_1, Q), e(C_1, Q)^s, H_1(e(C_1, Q)^s), m \rangle$ to the adversary.

Remark 6. In the hybrid system, TPM performs only two group exponentiation operations: one in G_1 and another in G_2. The RAM performs two group exponentiation operations in G_1, two group additions in G_1, three hash function evaluations, three bilinear map computations, and one XOR.

4.2. 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 $e(C_3, P) \stackrel{?}{=} e(hY + Z, C_1)$.
(3) $e(C_1, Q)^s = \alpha^r$, where $C_1 = rP$.

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

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

This proves the second assertion. Now,

$$U+V = \hat{h}C_1+C_4xC_1 = \hat{h}rP+txrP = r(\hat{h}P+txP) = r(\hat{h}P+tX)$$

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 $e(C_1, Q)^s = e(rP, Q)^s = [e(P, Q)^s]^r = \alpha^r$, Therefore,

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

$\texttt{KeyGen}^{\texttt{GB}}$

- Choose $x, s \in_R \mathbb{Z}_q$ and $Q, Y, Z \in_R \mathbb{G}_1$.
- Compute $X = xP \in \mathbb{G}_1$.
- Compute $\alpha = e(P, Q)^s \in \mathbb{G}_2$.
- The private key $\mathbf{sk} = \langle x, s \rangle \in \mathbb{Z}_q^2$ and the public key $\mathbf{pk} = \langle P, Q, X, Y, Z, \alpha \rangle \in \mathbb{G}_1^5 \times \mathbb{G}_2$.

$\texttt{Enc}^{\texttt{GB}}$

- 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 $C = \langle C_1, C_2, C_3, C_4 \rangle$.

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

4.3. Security

Theorem 1. *The encryption scheme* $\texttt{EncryptI}^{GB}$ *is IND-CCA2 secure with glass box decryption, if the DBDH assumption holds.*

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 $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against $\texttt{EncryptI}^{\texttt{GB}}$ scheme in the IND-CCA2 glass box decryption model with non-negligible advantage τ. We will show how to construct algorithm $\mathcal{C}$ that uses $\mathcal{A}$ to solve the DBDH problem in $\mathbb{G}_1$ and $\mathbb{G}_2$ with non-negligible advantage.

Setup: $\mathcal{C}$ generates the public key as follows:

- Set
$$P = R \tag{6}$$
- Set
$$Q = bR \tag{7}$$
- Set
$$\alpha = e(aR, bR) \tag{8}$$

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

Thus, the second component of the private key denoted as s is in fact a (implicitly). Now, choose $x \in_R \mathbb{Z}_q$ and set

$$X = xP \tag{9}$$

Thus, the private keys are $\langle x, s = a \rangle$.

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

$$\beta = \tilde{h}(cP) \tag{10}$$

$$h^* = H_3(\beta) \tag{11}$$

$$Y = \frac{1}{h^*}(Q + yP) \tag{12}$$

$$Z = -Q + \tilde{z}P \tag{13}$$

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

$\mathcal{C}$ now runs $\mathcal{A}_1$ with the generated public keys. $\mathcal{A}$ makes several queries to the glass box decryption oracle. $\mathcal{C}$ responds to queries made by $\mathcal{A}$ as follows.

$\mathcal{O}_{\texttt{Glass-Box-Dec}}$ Oracle: When a request for decryption of ciphertext $C = \langle C_1, C_2, C_3, C_4 \rangle$ is made, $\mathcal{C}$ decrypts it in the following way:

- Computes
$$\hat{h} = H_2(C_1, C_2) \tag{14}$$
$$U = \hat{h}C_1 \tag{15}$$

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

DOI: 10.1002/sec

Conventional System $\mathtt{Dec}^{\mathtt{GB}}$

- Compute $\hat{h} = H_2(C_1, C_2)$
- Compute $U = \hat{h}C_1$
- Compute $V = C_4 x C_1$
- Compute $h = H_3(U + V)$
- Check if $e(C_3, P) \stackrel{?}{=} e(hY + Z, C_1)$
  - If *true* then
    - Compute $m = C_2 \oplus H_1(e(C_1, Q)^s)$
  - Else
    - *ABORT*.

Hybrid System $\mathtt{Dec}^{\mathtt{GB}}$

- **NC**: Compute $\hat{h} = H_2(C_1, C_2)$
- **NC**: Compute $U = \hat{h}C_1$
- RAM→TPM: $\langle C_1, C_4 \rangle$
- **SC**: Compute $V = C_4 x C_1$
- TPM→RAM: V
- **NC**: Calculate $h = H_3(U + V)$.
- **NC**: Check if $e(C_3, P) \stackrel{?}{=} e(hY + Z, C_1)$

  If *true* then
  - **NC**: Compute $e(C_1, Q)$
  - RAM→TPM: $e(C_1, Q)$
  - **SC**: Compute $e(C_1, Q)^s$
  - TPM→RAM: $e(C_1, Q)^s$
  - **NC**: Compute $H_1(e(C_1, Q)^s)$
  - **NC**: Compute $m = C_2 \oplus H_1(e(C_1, Q)^s)$

  else *ABORT*.

$$V = C_4 x C_1 \tag{16}$$

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

$$h = H_3(U + V) \tag{17}$$

If $(h = h^*)$, then $\mathcal{C}$ aborts.[†] Because the Y and Z values are public, $\mathcal{C}$ computes correctly the value

$$hY + Z \tag{18}$$

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 $e(C_3, P) \stackrel{?}{=} e(hY+Z, C_1)$ passes, $\mathcal{C}$ must return the value $e(C_1, Q)^s$ 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 in terms of other quantities computable by $\mathcal{C}$. Observe that because P is a generator,

$$C_1 = rP, \text{ for some } r \in \mathbb{Z}_q \tag{19}$$

[†] We note that the probability of $\mathcal{C}$ aborting is negligible assuming the target collision resistance of H_3.

Because $e(C_3, P) = e(hY + Z, C_1)$, it follows that

$$C_3 = r(hY + Z) \tag{20}$$

for the same r defined in Equation (19). Now,

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

$\mathcal{C}$ 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 (12), (13), and (20),

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

Rearranging, we obtain

$$rQ = \left(\frac{h}{h^*} - 1\right)^{-1} \left[C_3 - \left(\frac{h}{h^*}y + \tilde{z}\right) C_1\right] \tag{21}$$

Observe that all values in the RHS of Equation (21) is available to $\mathcal{C}$. Thus, $e(C_1, Q)^s = e(aP, rQ)$ can be computed even without knowing s. Hence, the glass box decryption queries can be perfectly answered by $\mathcal{C}$ and return $\mathcal{I} = \langle \hat{h}, U, V, h, e(C_1, Q), e(C_1, Q)^s, H_1(e(C_1, Q)), m\rangle$ to $\mathcal{A}$. That is, $\mathcal{C}$ perfectly simulates the glass box decryption oracle to $\mathcal{A}$.

Challenge: $\mathcal{A}_1$ then outputs two messages m_0, m_1 of equal length. $\mathcal{C}$ computes the ciphertext C^* by performing the following steps:

- Set
$$C_1^* = cR = cP \tag{22}$$
cR is the input to the hard problem.
- Compute
$$C_2^* = m_\delta \oplus H_1(\gamma) \tag{23}$$
Here, $\delta \in \{0, 1\}$ is a random bit and γ is an input to the hard problem
- Compute
$$C_3^* = yC_1^* + \tilde{z}C_1^* \tag{24}$$
- Compute
$$C_4^* = (\hat{h} - \tilde{h})x^{-1} \tag{25}$$
Where, $\hat{h} = H_2(C_1^*, C_2^*)$ and $\tilde{h}$ was chosen by $\mathcal{C}$ at setup time. Note that 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 given as input to $\mathcal{A}_2$.

We first show that the challenge ciphertexts is a valid ciphertext. **Lemma 1** The challenge ciphertext $C^* = \langle C_1^*, C_2^*, C_3^*, C_4^*\rangle$ is a valid and properly formed ciphertext. □

Proof. Because $C_1^* = cP$, we should show that

$$C_3^* = c(hY + Z) \tag{26}$$

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

$$\begin{aligned} c(\hat{h}P + tX) &= c\left(\hat{h}P + C_4^* X\right) \\ &= c(\hat{h}P + (\hat{h} - \tilde{h})x^{-1}xP) \text{ (from Equation (25)} \\ &= c(\hat{h}P + \hat{h}P - \tilde{h}P) \\ &= \tilde{h}(cP) = \beta \text{ (from Equation (10))} \end{aligned}$$

Therefore,

$$h = H_3(c(\hat{h}P + tX)) = H_3(\beta) = h^* \tag{27}$$

From Equations (24) and (27), we conclude that $C^* = \langle C_1^*, C_2^*, C_3^*, C_4^*\rangle$ as defined previously will be a valid/consistent ciphertext if we show $C_3^* = c(h^*Y + Z)$. C_3^* was computed as $yC_1^* + \tilde{z}C_1^*$ in Equation (24). Thus, we have to show that

$$c(h^*Y + Z) = yC_1^* + \tilde{z}C_1^* \tag{28}$$

In fact,

$$\begin{aligned} c(h^*Y + Z) &= c[Q + yP - Q + \tilde{z}P] \text{ (from Equations (12)} \\ &\quad \text{and (13))} \\ &= y(cP) + \tilde{z}(cP) \\ &= yC_1^* + \tilde{z}C_1^* \end{aligned}$$

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

$\mathcal{C}$ answers $\mathcal{A}_2$'s glass box decryption oracle queries in an identical manner as described earlier.

Guess: Recall that the instance of the hard problem $\langle R, aR, bR, cR, \gamma\rangle$ is used by $\mathcal{C}$ as $P = R$, $Q = bR$, and $\alpha = e(a\hat{R}, bR) = e(P, Q)^s$ while setting the system. When constructing the challenge ciphertext, $C_1^* = cR = rP$ and $C_2^* = m_\delta \oplus H_1(\gamma)$. If the adversary $\mathcal{A}$ guesses the bit δ

correctly, then the challenger outputs 1. Else, it outputs 0. Now, we calculate the advantage $\mathcal{C}$ has in solving the DBDH problem. Recall that τ is the advantage the adversary has against the system. By definition of advantage, the probability that the adversary correctly guesses the bit that is encrypted is given by $1/2 + \tau/2$.

If γ is a DBDH instance, then the challenge ciphertext is distributed identically as the original ciphertext, and hence, the probability that the challenger outputs 1 is equal to $1/2 + \tau/2$. If γ is a random element from $\mathbb{G}_2$, then we show that the advantage the adversary has in distinguishing between $\delta = 0$ and $\delta = 1$ is negligible assuming the pseudorandomness property of H_1.

Claim 1. *If* $Pr[\delta = \delta' | \gamma = e(R,R)^d] = 1/2 + \epsilon$ *where* $d \in_R \mathbb{Z}_p$, *then* ϵ *is negligible assuming* H_1 *to a pseudorandom generator.*

Proof. We now construct an adversary $\mathcal{B}$ against the pseudorandom generator with advantage ϵ. $\mathcal{B}$ receives from the pseudorandom generator challenger a string y, which is either $H_1(\gamma)$ for a random γ or a uniformly chosen string of length l_m. $\mathcal{B}$ will play the security game exactly as the challenger described previously except for the following difference. In the challenge phase, it will set $C_2^* = m_\delta \oplus y$. $\mathcal{B}$ will output 1 if and only if $\mathcal{A}$ outputs $\delta' = \delta$. If $y = H_1(\gamma)$, then the probability that $\mathcal{B}$ outputs 1 is equal to $1/2 + \epsilon$. Otherwise, δ is information theoretically hidden because y is random. Hence, in this case, probability $\mathcal{B}$ outputs 1 is equal to 1/2. Therefore, the advantage $\mathcal{B}$ has against the pseudorandom generator is given by ϵ, which is negligible from the psuedo-randomness property of H_1. □

Thus, the advantage of $\mathcal{C}$ in solving the DBDH problem is given by

$$\begin{aligned} &|Pr[\mathcal{C} \text{ outp. } 1 | \gamma = e(R,R)^{abc}] - Pr[\mathcal{C} \text{ outp. } 1 | \gamma = e(R,R)^d]| \\ &\quad = (1/2 + \tau/2) - (1/2 + \epsilon) \\ &\quad = \tau/2 - \epsilon \end{aligned}$$

Because we have assumed τ to be non-negligible and we have just shown that ϵ is negligible, the advantage of $\mathcal{C}$ in solving the DBDH problem is non-negligible, which is a contradiction.

5. CONCLUSION

We have initiated the study of security under glass box decryption. We have shown that some of the most popular IND-CCA2 schemes are vulnerable to attacks in the glass box model even when minimal computation is done outside the TPM. We designed a new system having the nice property that minimal computations are done on the TPM and at the same time it is also secure under the glass box model.

REFERENCES

1. Baker W, *et al.* Data breach investigations report, 2010. (Available from: http://www.verizonbusiness.com/resources/reports/rp_2010-data-breach-report_en_xg.pdf) [accessed Jan 2011].
2. White SR, Comerford L. Abyss: an architecture for software protection. *IEEE Transactions on Software Engineering* 1990: 619–629.
3. Arvind A, Blanas S, Eguro K, Kaushik R, Kossmann D, Ramamurthy R, Venkatesan R. Orthogonal security with cipherbase. In *CIDR 2013, Sixth Biennial Conference on Innovative Data Systems*. Research, Asilomar: CA, USA, 2013.
4. Alwen J, Dodis Y, Wichs D. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *Advances in Cryptology - CRYPTO 2009*, vol. 5677, Lecture Notes in Computer Science. Springer: Santa Barbara, 2009.
5. Barak B, Goldreich O, Impagliazzo R, Rudich S, Sahai A, Vadhan S, Ke Y. On the (im) possibility of obfuscating programs. In *Advances in CryptologyCRYPTO 2001*. Springer: Santa Barbara, 2001; 1–18.
6. Garg S, Gentry C, Halevi S, Raykova M, Sahai A, Waters B. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*. IEEE: Berkeley, 2013; 40–49.
7. Apon D, Huang Y, Katz J, Malozemoff AJ. Implementing cryptographic program obfuscation. *IACR Cryptology ePrint Archive*, 2014; 779.
8. Fujisaki E, Okamoto T. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference*, vol. 1666, Lecture Notes in Computer Science. Springer: Santa Barbara, 1999; 537–554.
9. Cramer R, Shoup V. A practical public key cryptosystem probably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology - CRYPTO '98*, vol. 1462, Lecture Notes in Computer Science. Springer: Santa Barbara, 1998; 13–25.
10. Goldreich O. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.
11. El Gamal T. On computing logarithms over finite fields. In *Advances in Cryptology - CRYPTO 1985*, vol. 218, Lecture Notes in Computer Science. Springer: Santa Barbara, 1985; 396–402.
12. Kiltz E, Malone-Lee J. A general construction of IND-CCA2 secure public key encryption. In *Cryptography and Coding, 9th IMA International Conference*, vol. 2898, Lecture Notes in Computer Science. Springer: Cirencester, UK, 2003; 152–166.

13. Fujisaki E, Okamoto T. 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*, vol. 1560, Lecture Notes in Computer Science. Springer, 1999.

APPENDIX A: CRAMER–SHOUP CRYPTOSYSTEM

We now show how the well known Cramer-Shoup system which is CCA2 secure can be broken if the black box decryption oracle is replaced with a glass box decryption oracle. We review the Cramer-Shoup encryption [9] scheme.

- $\mathcal{CS}$.Gen: The private key and public key of a user are $sk = (x_1, x_2, y_1, y_2, z_1, z_2)$ and public key $pk = (g_1, g_2, c, d, h)$, where $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$ and $h = g_1^{z_1} g_2^{z_2}$.
- $\mathcal{CS}$.Enc: Compute $u_1 = g_1^r$, $u_2 = g_2^r$, $e = h^r m$, $\alpha = H(u_1, u_2, e)$ and $v = c^r d^{r\alpha}$. $C = \langle u_1, u_2, e, v \rangle$.
- $\mathcal{CS}$.Dec: We explain the implementation of the decryption algorithm using a s module and using TPM separately. We stress that there may be an implementation of the Cramer-Shoup encryption scheme, where all the steps of decryption algorithm are computed in the TPM, which gives no greater advantage to the glass box decryption than the black box decryption oracle. In the TPM based implementation under our consideration, we do not perform any computation which involves the secret key outside the TPM. Still we are able to mount glass box attack on the implementation. On receiving a ciphertext $C = \langle u_1, u_2, e, v \rangle$ decryption is done as follows:

Consider the glass box execution of Decryption oracle on a ciphertext (u_1, u_2, e, v),

(a) Since all these are input parameters, these are visible/available to the adversary.

Conventional System:

- Compute $\alpha = H(u_1, u_2, e)$.
- Compute $V = u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^{\alpha}$.
- If $(v = V)$ then,
  - Compute $Z = u_1^{z_1} u_2^{z_2}$.
  - Compute $m = e/Z$ and return m.

Else *ABORT*

(b) In the evaluation of the expression $\alpha = H(u_1, u_2, e)$ all values will be available to the adversary.

(c) However, the expression $V = u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^{\alpha}$ is evaluated using the TPM because this involves secret keys x_1, x_2, y_1, y_2. Thus, u_1, u_2 and α are sent to the TPM and the value $V = u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^{\alpha}$ is sent to the normal world and hence V is available to the adversary.

(d) The check $(v \stackrel{?}{=} V)$ is done outside the TPM. If this fails the algorithm is aborted and adversary gets no further values. If $(v = V)$ is true, then the TPM is used to obtain the value of $Z = u_1^{z_1} u_2^{z_2}$ and Z is sent outside the TPM. Now, the adversary obtains the values Z and $m = e/Z$ as well.

(e) Therefore the set $\mathcal{I} = \langle \alpha, V, Z, m \rangle$.

We will now show how an adversary having access to glass box decryption can distinguish challenge messages. During the challenge phase $\mathcal{A}$ selects two messages $\{m_0, m_1\}$ and sends them to $\mathcal{C}$. Now, $\mathcal{C}$ constructs the challenge ciphertext C^* as $C^* = \langle u_1^*, u_2^*, e^*, v^* \rangle = \langle u_1, u_2, (u_1)^{z_1} (u_2)^{z_2} m_\delta, (u_1)^{x_1} (u_2)^{x_2} \ ((u_1)^{y_1} (u_2)^{y_2})^{\alpha} \rangle$, where δ is a random bit $\in \{0, 1\}$ and $\alpha = H(u_1^*, u_2^*, e^*)$. The challenger sends C^* to $\mathcal{A}$ and asks him to find the m_δ hidden in C^*. At this point the *second phase* of the training begins and $\mathcal{C}$ must respond to all legal queries raised by $\mathcal{A}$. This is what $\mathcal{A}$ asks to find m_δ.

- $\mathcal{A}$ chooses $s_1 \in_R \mathbb{Z}_q^*$ and constructs a ciphertext $C' = \langle u_1', u_2', e', v' \rangle = \langle (u_1^*)^{s_1}, (u_2^*)^{s_1}, e^*, v^* \rangle$, where u_1^* and u_2^* are the first two components of C^*. In other words C' is nothing but C^* with the first two components, namely u_1^* and u_2^* exponentiated with s_1.
- Now, $\mathcal{A}$ queries Glass-Box-Dec(C'). Note that it is legal to ask the decryption of C'.
- As $\mathcal{C}$ knows all the private keys, it would faithfully execute the $\mathcal{CS}$.Dec on C'.
- $\mathcal{C}$ will reject the ciphertext C' because $v' \neq (u_1')^{x_1} (u_2')^{x_2} ((u_1')^{y_1} (u_2')^{y_2})^{\alpha_1}$.

Hybrid System:

- NC: Compute $\alpha = H(u_1, u_2, e)$.
- RAM→TPM: $\langle \alpha, u_1, u_2 \rangle$
- SC: Compute $V = u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^{\alpha}$.
- TPM→RAM: V
- NC: If $(v = V)$ then,
  - SC: Compute $Z = u_1^{z_1} u_2^{z_2}$.
  - TPM→RAM: Z
  - NC: Compute $m = e/Z$ and return m.

Else *ABORT*

DOI: 10.1002/sec

- Now, $\mathcal{I}=\langle\alpha_1, V_1, Z, m\rangle = \langle H(u_1', u_2', e'), (u_1^*)^{s_1x_1} (u_2^*)^{s_1x_2}((u_1^*)^{s_1y_1}(u_2^*)^{s_1y_2})^{\alpha_1}, -, -\rangle$.
- Similarly, $\mathcal{A}$ constructs another ciphertext C'' by choosing $s_2 \in_R \mathbb{Z}_q^*$, computing $u_1'' = (u_1^*)^{s_2}$, $u_2'' = (u_2^*)^{s_2}$, $e'' = e^*$ and $v'' = v^*$. The newly formed ciphertext is $C'' = \langle u_1'', u_2'', e'', v''\rangle$ $\mathcal{A}$ queries `Glass -Box-Dec`(C'').
- $\mathcal{C}$ will reject C'' because it is invalid.
- Here, $\mathcal{I}=\langle\alpha_2, V_2, Z, m\rangle = \langle H(u_1'', u_2'', e''), (u_1^*)^{s_2x_1} (u_2^*)^{s_2x_2}((u_1^*)^{s_2y_1}(u_2^*)^{s_2y_2})^{\alpha_2}, -, -\rangle$.

We will now show that with the values V_1 and V_2, $\mathcal{A}$ performs the following and obtains m_δ:

- Computes $X_1 = V_1^{s_1^{-1}} = (u_1^*)^{x_1}(u_2^*)^{x_2}((u_1^*)^{y_1}(u_2^*)^{y_2})^{\alpha_1}$ and $X_2 = V_2^{s_2^{-1}} = (u_1^*)^{x_1}(u_2^*)^{x_2}((u_1^*)^{y_1}(u_2^*)^{y_2})^{\alpha_2}$.
- Computes $Y = \frac{X_1}{X_2} = ((u_1^*)^{y_1}(u_2^*)^{y_2})^{\alpha_1-\alpha_2}$.
- Computes $Z_2 = Y^{(\alpha_1-\alpha_2)^{-1}} = (u_1^*)^{y_1}(u_2^*)^{y_2}$.
- Computes $Z_1 = \frac{X_1}{Z_2^{\alpha_1}} = (u_1^*)^{x_1}(u_2^*)^{x_2}$.
- Generates a fresh ciphertext by computing $\hat{u}_1 = u_1^*$, $\hat{u}_2 = u_2^*$, $e = e^*\hat{m}$ and $\hat{v} = Z_1 Z_2^{\hat{\alpha}}$, where $\hat{m}$ is an arbitrary message chosen by $\mathcal{A}$ and $\hat{\alpha} = H(\hat{u}_1, \hat{u}_2, e)$.
- Now, $\hat{C} = \langle\hat{u}_1, \hat{u}_2, e, \hat{v}\rangle$ is a valid encryption on message $m_\delta\hat{m}$ (Due to Lemma 2) and different from C^*. Thus $\mathcal{A}$ can legally query `Glass-Box-Dec`$(\hat{C})$.
- $\mathcal{C}$ returns $(u_1^*)^{x_1}(u_2^*)^{x_2}((u_1^*)^{y_1}(u_2^*)^{y_2})^{\hat{\alpha}}$ and $m_\delta\hat{m}$ as the output (Due to Lemma 2).
- Since $\mathcal{A}$ knows the value $\hat{m}$, $\mathcal{A}$ can easily obtain the message m_δ from $(m_\delta\hat{m})$.
- Thus, $\mathcal{A}$ identifies the bit δ almost always.

Lemma 2. *The ciphertext $\hat{C} = \langle\hat{u}_1, \hat{u}_2, e, \hat{v}\rangle$ is a valid ciphertext and the glass box decryption returns $\mathcal{I}=\langle\hat{\alpha}, V, Z, m\rangle = \langle\hat{\alpha}, (u_1^*)^{x_1}(u_2^*)^{x_2}((u_1^*)^{y_1}(u_2^*)^{y_2})^{\hat{\alpha}}, \hat{u}_1^{z_1}\hat{u}_2^{z_2}, m_\delta\hat{m}\rangle$ as the output.*

Proof. The ciphertext $\hat{C} = \langle\hat{u}_1, \hat{u}_2, e, \hat{v}\rangle = \langle u_1^*, u_2^*, e^*\hat{m}, Z_1 Z_2^{\hat{\alpha}}\rangle$. $\mathcal{C}$ checks whether $\hat{C}$ is valid by performing the check $\hat{v} \stackrel{?}{=} (\hat{u}_1)^{x_1}(\hat{u}_2)^{x_2}$ $((\hat{u}_1)^{y_1}(\hat{u}_2)^{y_2})^{\hat{\alpha}}$, where $\hat{\alpha} = H(\hat{u}_1, \hat{u}_2, e)$. Below we show that $\hat{C}$ passes this verification:

$$\begin{aligned} RHS &= (\hat{u}_1)^{x_1}(\hat{u}_2)^{x_2}\left((\hat{u}_1)^{y_1}(\hat{u}_2)^{y_2}\right)^{\hat{\alpha}} \\ &= (u_1^*)^{x_1}(u_2^*)^{x_2}\left(\left(u_1^*\right)^{y_1}(u_2^*)^{y_2}\right)^{\hat{\alpha}} \\ &= Z_1(Z_2)^{\hat{\alpha}} \\ &= \hat{v} = LHS \end{aligned}$$

Since the above check returns true, $\mathcal{C}$ performs the decryption by computing $e/(\hat{u}_1)^{z_1}(\hat{u}_2)^{z_2})$. We show that this computation outputs $\hat{m}m_\delta$:

$$\begin{aligned} \frac{e}{(\hat{u}_1)^{z_1}(\hat{u}_2)^{z_2}} &= \frac{e^*\hat{m}}{(\hat{u}_1)^{z_1}(\hat{u}_2)^{z_2}} = \frac{(u_1)^{z_1}(u_2)^{z_2}m_\delta\hat{m}}{(\hat{u}_1)^{z_1}(\hat{u}_2)^{z_2}} \\ &= \frac{(u_1^*)^{z_1}(u_2^*)^{z_2}m_\delta\hat{m}}{(u_1^*)^{z_1}(u_2^*)^{z_2}} = m_\delta\hat{m} \end{aligned}$$

Since $u_1^* = \hat{u}_1 = u_1$ and $u_2^* = \hat{u}_2 = u_2$ □

Remark 7. Notice that only one step is computed outside TPM but the value exposed due to that is sufficient for the adversary to break the system.

APPENDIX B: KILTZ-MALONE LEE TRANSFORMATION

We review the Kiltz-Malone Lee transform given in [12] and show the glass box attack of the same here. Given a weak symmetric key encryption scheme $\mathtt{Enc}^{sym}$ (Weak in the sense that, the adversary cannot distinguish the encryption of two messages m_1 and m_2 without access to encryption or decryption oracle), where K is the key for the symmetric key encryption scheme and M is the message to be encrypted), it is possible to construct a CCA2 secure encryption scheme by choosing a random r from the appropriate space and computing $h = H(m\|r)$. Now, the encryption of a message m is:

$$\mathtt{Enc}_{pk}^{\mathtt{CCA2}}(m) = c = \langle c_1, c_2\rangle = \langle(f_1(h), \mathtt{Enc}_{G(f_2(h))}^{sym}(m\|r))\rangle$$

Where,

- The functions $f_1(.)$ and $f_2(.)$ are functions such that given h, finding $f_1(h)$ and $f_2(h)$ are easy but given $f_1(h)$ alone, finding $f_2(h)$ is intractable. Named as the Y – *Computational* problem in [12].
- $H(.)$ and $G(.)$ are two hash functions with appropriate domain and co-domains, considered as random oracles.
- $\mathtt{Enc}_{G(f_2(h))}^{sym}(m\|r)$ is the symmetric key encryption of $m\|r$ with the hash of $f_2(h)$ as the key.

Decryption is done as follows:

- Compute $k = G(t(c_1))$, $m'\|r' = \mathtt{Dec}_k^{sym}(c_2)$ and $h' = H(m'\|r')$.
- Accept m' only if $c_1 = f_1(h'))$

Where, t is a function which satisfies $t(f_1(x)) = f_2(x)$.

B.1 Glass Box Attack on Kiltz-Malone Lee Transformation:

Let Δ_{CCA2} be a CCA2 secure encryption scheme instantiated by using the Kiltz-Malone Lee Transformation. Let $\mathcal{C}$ be the challenger simulating the instance Δ_{CCA2} using glass box decryption oracle and $\mathcal{A}$ be the adversary who

is assumed to break the confidentiality of Δ_{CCA2}. $\mathcal{A}$ can distinguish the ciphertext as follows:

- After the training phase, $\mathcal{A}$ sends two messages (m_0, m_1) to $\mathcal{C}$.
- $\mathcal{C}$ sends back the challenge ciphertext $c^* = \langle c_1^*, c_2^* \rangle = \langle (f_1(h), \mathtt{Enc}^{sym}_{G(f_2(h))}(m_\delta \| r)) \rangle$, where $\delta \in_R \{0, 1\}$ to $\mathcal{A}$.
- $\mathcal{A}$ cooks up a new ciphertext $c' = \langle c_1', c_2' \rangle$, where $c_1' = c_1^*$ and $c_2' \in_R Range(\mathtt{Enc}^{sym})$, i.e. a random element in the range of the output of the symmetric key encryption scheme.
- $\mathcal{A}$ queries `Glass-Box-Dec`(c').
- $\mathcal{C}$ returns $\mathcal{I} = \langle G(t(c_1')), m' \| r', h' \rangle = \langle G(t(c_1^*)), \mathtt{Dec}^{sym}_k(c_2), H(m' \| r') \rangle$ as the values computed outside the TPM, but rejects the ciphertext because the check $c_1 \stackrel{?}{=} f_1(h'))$ will fail (since the ciphertext component c_2' formed by $\mathcal{A}$ is not meaningful).
- $\mathcal{A}$ makes use of k' which is same as the k^* used for generating the challenge ciphertext and computes $m_\delta = \mathtt{Dec}^{\mathtt{Sym}}_{G(k')}(c_2^*)$.

Thus, an adversary who has access to the glass box decryption oracle can break the indistinguishability of Kiltz-Malone Lee Transformation.