An Identity Based Encryption Scheme Resilient to RAM Scraper Like Malware Attacks

Dipanjan Das, Priyanka Bose, S. Sree Vivek,
S. Sharmila Deva Selvi, and C. Pandu Rangan

Theoretical Computer Science Laboratory,
Department of Computer Science and Engineering,
Indian Institute of Technology, Madras
Chennai, India.
{priyab,dipanjan,svivek,sharmila,rangan}@cse.iitm.ac.in

Abstract. Modern software ecosystem is data-centric. Data exfiltration due to the attacks of Memory Scraper type malwares is an emerging threat. In this paper, we set up an appropriate mathematical model capturing the threat such attacks pose to Identity Based Cryptosystems (IBE). Following the formalism, we demonstrate an attack on popular Boneh-Franklin CCA2 secure IBE construction that compels us to *relook* the fact of CCA2 being the *de-facto* standard of security. We offer two constructions, one identity based and another public-key based (PKE) encryption schemes capable of withstanding Ram Scraper attacks. Our design assumes a hybrid system equipped with a bare minimal 'Trusted Platform Module' (TPM) that can *only* perform group exponentiation operation. Building systems to implement our IBE/PKE protocols should be feasible as well as efficient from practical standpoint.

Keywords: Glassbox Security Model, Identity Based Encryption, RAM Scraper, Trusted Platform Module, Hybrid System, Malware

1 Introduction

Over the last decade, the notion of CCA2 security remained to be the highest standard that can be achieved by an encryption algorithm. In this model of security, crucial point is that the challenger offers to an adversary is the decryption of chosen ciphertext. However, recent trend has shown the emergence of well-crafted and sophisticated attacks which expose much more information to the adversary, *e.g.* certain number of bits of secret key and intermediate values generated during the execution of cryptographic algorithms.

In contrast to traditional cryptanalysis that views the cryptosystem as a 'black' *box* accepting input and producing output; side channel attacks peeps into the same *box* to retrieve the secret key which the decryption process is parameterized on. Side channel cryptanalysis targets the implementation aspects rather than the algorithm itself to aid in key-recovery. In [18], Yarom *et al.* have demonstrated how they could mount a Flush+Reload attack by exploiting a weakness in Intel x86 processors to recover 96.7% of the bits of the secret key of a victim program running GnuPG v1.4.13 by observing a single signature or decryption round. Leakage resilient cryptography is concerned with modeling this kind of threat.

One immediate solution that crosses our mind to tackle the issue above is to keep the secret key safeguarded inside a tamper resistant hardware module. Intuitively, it might seem to be a plausible solution to the problem at hand, but Ram Scrapers have taken the threat one-step further. It is a piece of data-harvesting malware [8] that collects data from volatile memory. Rather than taking a whole memory snapshot, those often use stealthy techniques, such as, hooking into a payment processing application and selectively dumping data that matches certain patterns, *e.g.* the regular expression of a credit card format from a specific memory region. Ever since VISA alerted us [9] to Ram Scrapers back in 2008, point-of-sale (PoS) terminals have become a 'juicy' attack vector for such malwares [Fig.1]. Though encryption protects data during 'transit' or at 'rest', Ram Scraper exposes data to prying eyes during 'processing' while those stay in unencrypted state in system memory. It becomes even severe when the intermediate values are leaked in the course of an execution of a cryptographic algorithm enabling an attacker successfully decrypt ciphertexts encrypted in the past. [Sec.5.2] demonstrates one such attack against a CCA2 [Sec.4.5] secure IBE [Sec.4.4] by taking into account the extra amount of information an adversary would obtain from memory scraping. We prefer to call this enhanced model of security as 'Glassbox for IBE' [Sec.4.5]. In our theoretical framework, the adversary, that mimics the behavior of a Ram Scraper, is armed with a Glassbox decryption oracle $\mathcal{O}_{\texttt{GB-ID}}$ which provides those additional values during simulation.

In order to circumvent such sophisticated attacks, cryptographic algorithms are often deployed to a 'Hybrid System' [Fig.2] having a tamper-resistant 'Trusted Platform Module' (TPM) installed. A TPM is a system component that has state separate from the system on which it reports (the host system). In [16], Trusted Computing Group (TCG) has defined TPM v2.0 to be equipped with asymmetric engine(s), symmetric engine(s), hash engine(s), random number generator, execution engine(s), management, authorization, key generation and power detection modules - all connected to a single data communication path. Operations supported by cryptographic subsystem includes hash computation, asymmetric and symmetric encryption/decryption, signing/verification and key generation. TPM has become so popular a piece of device for trusted computing that newer versions of latest motherboards support dedicated slots for TPM to be plugged in. BitLocker drive encryption [12], a new security feature that ships with latest versions of Windows operating system, uses the TPM to lock the encryption keys that protect the data. As a result, the keys cannot be accessed until the TPM has verified the state of the computer. Because the keys needed to decrypt data remain locked by the TPM, an attacker cannot read the data just by removing the hard disk and installing it in another computer. Secret key or its components are safeguarded inside TPM and the values computed in it are not available to the attacker. However, TPMs have very limited memory and processing power. Hence, we assume that our protocol is partly executed in TPM and the rest of the steps in insecure computing environment. We assume the existence of a minimal TPM to run our protocol. To be precise, only requirement for the TPM is to be able to perform group exponentiation. However, looking into future one might argue that efficient implementation of specifications like TinyPBC [1] can make embedded processors perform pairing, but it should never be

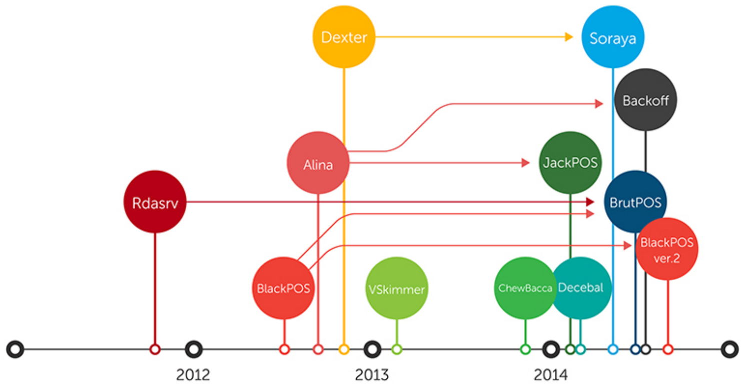

Fig. 1. PoS RAM SCRAPER family tree. © 2011 Trend Micro Incorporated. All Rights Reserved.

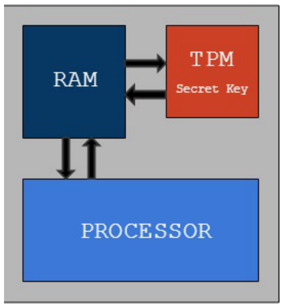

Fig. 2. Interface among Processor, RAM and TPM in a Hybrid System

forgot that pairing is inherently a costly computation. Our design goal was to come up with a simplistic protocol not needing any specialized support for any mathematical routine, *e.g* elliptic curve operations. Therefore, we have employed a clever tweak to reduce the requirements on a TPM and pushing pairing in RAM, thus offloading the TPM.

In [17], Vivek *et al.* discusses such a system as above, but in PKI setting. IBE inherently relieves the user from the hassle of certificate management in PKI system. In this paper, we re-examine IBE schemes to offer a Glassbox-proof construction addressing the threats posed by MEMORY SCRAPER type malwares.

2 Related Works

The emergence of side channel attacks have lead us to designing completely novel secure cryptographic systems. In [2], Akavia *et al.* extended the security model by designing schemes against freezing attacks where the adversary can gather significant amount of information about secret key from a part of memory that can be accessed even if the power is off. The leakage resilient cryptography has also played a very important role in designing secure protocols [10][3][11]. It should be noted that though the above schemes are designed mainly for key leakage resilience, the effect of RAM/MEMORY SCRAPERS to all these systems can be disastrous. The idea of such kind of attacks first appeared in [14]. Here the authors have proposed a new security model (seCK) for authenticated key agreement and talks about MEMORY SCRAPERS which can grab information from volatile memory except those in tamper resilient device. In [17], Vivek *et al.* introduced first PKE scheme provably secure both in standard and random oracle models against MEMORY SCRAPERS. But, formalizing such a notion for IBE has not been explored till date.

3 Our Contribution

Our contribution is two-fold. In this paper, we have provided a theoretical framework for the real-world attacks launched by MEMORY SCRAPER like malwares. Modeling and analysis of similar threats have already been researched in the domains of key-exchange and public-key cryptosystem [17], but our paper extends and introduces that concept for identity based system for the first time. We call our security notion as "Adaptive Chosen Ciphertext Attack Security with Glassbox Decryption for Identity Based Cryptosystems" (`GB-ID`). Also we illustrate the relevance of our security model by demonstrating the vulnerability in Boneh-Franklin CCA2 secure IBE scheme [5]. It emphasizes our claim `IND-ID-GB` security to be a stronger requirement than CCA2 compliance. We have designed both a PKE and an IBE system secure in our proposed model. The protocols are aimed to be simple yet practical ones, thus executing only secret-key involving computations inside Trusted Platform Module (TPM) and remaining ones in insecure volatile storage, e.g. Random Access Memory (RAM) etc.

4 Preliminaries

4.1 Notations

By PPT, we mean a probabilistic polynomial time algorithm with respect to a security parameter κ. All adversaries defined here will be PPT except stated otherwise. Given a probability distribution $\mathcal{D}$ and an element y, $y \leftarrow \mathcal{D}$ denotes selecting an element y according to $\mathcal{D}$. Let $\mathcal{A}$ be a probabilistic algorithm , then $\mathcal{A}(x_1...x_n)$ describes the output distribution of $\mathcal{A}$ based on inputs $x_1, x_1, ..., x_n$. We use $\overline{E}$ to denote the complement of some event E. $\langle G \rangle$ denotes the group generated by the group generator G. $\mathbb{Z}_p$ denotes set of all integers modulo p, where p is prime. A function $\nu : \mathbb{N} \rightarrow \mathbb{R}^+$ is said to be negligible if $\forall c > 0$, $\exists k'$ such that $\nu(k) < k^{-c}$ for all $k' < k$.

4.2 Bilinear Groups

A bilinear pairing defined to be $\mathcal{G} = (p, \mathbb{G}_a, \mathbb{G}_b, \mathbb{G}_t, e, P_a, P_b)$ where we choose $\mathbb{G}_a = \langle P_a \rangle$, $\mathbb{G}_b = \langle P_b \rangle$ as additive and $\mathbb{G}_t$ as multiplicative groups of prime order p. A bilinear pairing e is a map $e : \mathbb{G}_a \times \mathbb{G}_b \rightarrow \mathbb{G}_t$ having the following properties.

- ***Bilinearity***: For $P_a \in \mathbb{G}_a$, $P_b \in \mathbb{G}_b$ and $x, y \in \mathbb{Z}_p$ the following holds true: $e(P_a^x, P_2^y) = e(P_a, P_b)^{xy}$.
- ***Non-degeneracy***: For any $\mathcal{X} \in \mathbb{G}_a$ and $\mathcal{Y} \in \mathbb{G}_b$, if $e(\mathcal{X}, \mathcal{Y}) = 1_T$, the identity element of $\mathbb{G}_t$, then either $\mathcal{X}$ is the identity of $\mathbb{G}_a$ or $\mathcal{Y}$ is the identity of $\mathbb{G}_b$.
- ***Efficiently Computable***: The map e should be efficiently computable.

Here we will use $\mathbb{G}_a = \mathbb{G}_b = \mathbb{G}_1, \mathbb{G}_a = \langle P_1 \rangle$ and $\mathbb{G}_t = \mathbb{G}_2$.

4.3 Computational Bilinear Diffie-Hellman Assumption (CBDH)

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be groups of prime order p. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be an admissible bilinear map and $\mathbb{G}_1 = \langle P_1 \rangle$. The **CBDH** problem in $(\mathbb{G}_1, \mathbb{G}_2, e)$ is defined as: given $\langle P_1, aP_1, bP_1, cP_1 \rangle$ where $a, b, c \in \mathbb{Z}_p^*$, compute $\mathbb{W} = e(P_1, P_1)^{abc} \in \mathbb{G}_2$. **CBDH** is assumed to be hard if for all PPT adversaries $\mathcal{A}$,

$$|\Pr[\mathcal{A}_{BDH}(P_1, aP_1, bP_1, cP_1) = e(P_1, P_1)^{abc}] \leq \nu(\kappa)$$

4.4 Identity Based Encryption (IBE)

An identity-based encryption scheme, first introduced in [15], is defined by the following four algorithms:

- **Setup**(κ): It takes the security parameter κ and returns systems parameters $\texttt{params}$ (announced to public) and master secret key (MSK) (known to private key generator (PKG) only). The system parameters include finite message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$.
- **KeyGen**$(\texttt{params}, \texttt{MSK}, \texttt{ID})$: It takes $\texttt{params}$, $\texttt{MSK}$ and an arbitrary identity $\texttt{ID} \in \{0, 1\}^*$ as input and returns a private key $d_{\texttt{ID}}$. Here, $\texttt{ID}$ is used as the public key and $d_{\texttt{ID}}$ as secret key.
- **Encrypt**$(\texttt{params}, \texttt{ID}, M)$: It takes $\texttt{params}$, $\texttt{ID}$ and a message $M \in \mathcal{M}$ as input and returns the ciphertext $C \in \mathcal{C}$.
- **Decrypt**$(\texttt{params}, C, d_{\texttt{ID}})$: It takes $\texttt{params}$, $C \in \mathcal{C}$, a the private key $d_{\texttt{ID}}$ as input and returns a message $M \in \mathcal{M}$.

The above four algorithm should satisfy the standard consistency check constraint i.e $\forall M \in \mathcal{M} : \texttt{Decrypt}(\texttt{params}, \texttt{Encrypt}(\texttt{params}, \texttt{ID}, M), d_{\texttt{ID}}) = M$

4.5 Security Models

Chosen Ciphertext Attack (IND-ID-CCA2) Security: CCA2 is the standard acceptable notion of security for public key encryption (PKE) schemes [4][13]. Therefore, the same has been adapted in IBE [5] to incorporate this stronger notion of security. In an IBE scheme, CCA2 security game (IND-ID-CCA2) is first defined in [5]. An identity-based encryption scheme is said to be semantically secure against chosen ciphertext attack (IND-ID-CCA2) if an PPT adversary $\mathcal{A}$ has an negligible advantage in the following game:

- **Setup**: The challenger $\mathcal{C}$ takes security parameter κ as input and runs the $\texttt{Setup}$ algorithm. It provides $\mathcal{A}$ the system parameters $\texttt{params}$ and keeps the master secret key (MSK) to itself.
- **Phase 1**: $\mathcal{A}$ makes queries $q_1, ..., q_n$ where query q_i is either of the following:
  - Extraction query $\langle \texttt{ID}_i \rangle$: $\mathcal{C}$ runs $\texttt{Extract}$ and generates the private key d_i corresponding to the public key $\texttt{ID}_i$. It gives d_i to $\mathcal{A}$.

  - Decryption query $\langle C_i, \mathtt{ID}_i \rangle$: $\mathcal{C}$ first runs `Extract` to generate the private key d_i for $\langle \mathtt{ID}_i \rangle$ and using d_i, it runs the algorithm `Decrypt` to decrypt the ciphertext C_i. It sends the message to $\mathcal{A}$.

  These queries may be adaptive, *i.e.* each query q_i may depend on the previous replies to $q_1, ..., q_{i-1}$

- **Challenge**: Once $\mathcal{A}$ decides that Phase 1 training is over, it outputs two messages $M_0, M_1 \in \mathcal{M}$ of equal length and an identity $\mathtt{ID}^*$ which it wishes to be challenged on. The only constraint is that $\mathtt{ID}^*$ did not appear in any of the private key extraction queries in Phase 1. Then $\mathcal{C}$ randomly chooses $\gamma \in_R \{0, 1\}$ and sets $C^* = \mathtt{Encrypt}(\mathtt{params}, \mathtt{ID}^*, M_\gamma)$. It sends the challenge ciphertext C^* to $\mathcal{A}$.
- **Phase 2**: $\mathcal{A}$ asks more queries $q_{n+1}, ...q_m$ where q_i is one of the following:
  - Extraction query $\langle \mathtt{ID}_i \rangle$: This queries are same as Phase 1 queries and they may be asked adaptively. Only constraint is that $\mathtt{ID}^* \neq \mathtt{ID}_i$.
  - Decryption query $\langle C_i, \mathtt{ID}_i \rangle$: This queries are similar to Phase 1 queries except $\langle C_i, \mathtt{ID}_i \rangle \neq \langle C^*, \mathtt{ID}^* \rangle$.
- **Guess**: At the end, $\mathcal{A}$ outputs $\gamma' \in \{0, 1\}$. $\mathcal{C}$ outputs 1 if $\gamma = \gamma'$ ($\mathcal{A}$ wins the game), else outputs 0.

We define adversary $\mathcal{A}$'s advantage against the CCA2 security of an IBE scheme as:

$$Adv_{\mathcal{A}}^{\mathcal{E},\mathtt{ID\text{-}CCA2}} = |Pr(\gamma = \gamma') - \frac{1}{2}| \leq \nu(\kappa)$$

Glassbox-Decryption Oracle for IBE: ($\mathcal{O}_{\mathtt{GB\text{-}ID}}$): In Glassbox security game for IBE, the challenger substitutes standard CCA2 decryption oracle for IBE ($\mathcal{O}_{\mathtt{CCA2\text{-}ID}}$) with this more *verbose* version of the same what we call as Glassbox-Decryption Oracle for IBE ($\mathcal{O}_{\mathtt{GB\text{-}ID}}$). Simulation of $\mathcal{O}_{\mathtt{GB\text{-}ID}}$ is only relevant in a hybrid setting with the IBE decryption algorithm precisely partitioned to execute partially in both RAM and TPM. Consider the decryption oracle trying to recover the plaintext M from the ciphertext C encrypted under the identity ID having the secret key $d_{\mathtt{ID}}$. While $\mathcal{O}_{\mathtt{CCA2\text{-}ID}}$ hands over to the adversary either the correct decryption M or an **ABORT** message in case of incorrect decryption; in addition to that $\mathcal{O}_{\mathtt{GB\text{-}ID}}$ empowers the adversary by disclosing a set of intermediate values $\mathcal{I} \leftarrow \mathcal{O}_{\mathtt{GB\text{-}ID}}(C, \mathtt{ID}, d_{\mathtt{ID}})$ which represents the collection of all the entities computed/stored in RAM (outside TPM) until the decryption algorithm of IBE protocol prematurely aborts/successfully terminates.

Glassbox (IND-ID-GB) Security: In [17], Vivek *et al.* has introduced the notion of Glassbox security for public-key based cryptosystems by allowing the adversary access to Glassbox-Decryption oracle for PKE ($\mathcal{O}_{\mathtt{GB\text{-}PK}}$). We adapt this notion further to IBE framework by replacing the regular black box decryption oracle as present in CCA2 game ($\mathcal{O}_{\mathtt{CCA2\text{-}ID}}$) with a flexible Glassbox-Decryption oracle for IBE ($\mathcal{O}_{\mathtt{GB\text{-}ID}}$). An identity-based encryption scheme is said to be secure against Glassbox attack (IND-ID-GB) if an PPT adversary $\mathcal{A}$ has an negligible advantage in the following game:

- **Setup**: Identical to CCA2 security game.
- **Phase 1**: Key extraction query is same as CCA2 security. The only difference is that the adversary will get access to the $\mathcal{O}_{\texttt{GB-ID}}$ instead of ordinary black box CCA2 decryption oracle.
- **Challenge**: The challenger constructs the challenge ciphertext C^* on $\texttt{ID}^*$ by encrypting a random message $M_\gamma, \gamma \in_R \{0,1\}$ in the same way as in the CCA2 security game with a condition that $\texttt{ID}^*$ did not appear in any on the private key extraction query.
- **Phase 2**: Same as Phase 1. In case of Key extraction query on any $\texttt{ID}$, $\texttt{ID} \neq \texttt{ID}^*$. $\mathcal{A}$ will have access to $\mathcal{O}_{\texttt{GB-ID}}$ with a condition that $\langle C^*, \texttt{ID}^* \rangle \neq \langle C, \texttt{ID} \rangle$ for decryption query on $\langle C, \texttt{ID} \rangle$.
- **Guess** $\mathcal{A}$ will output γ'. $\mathcal{C}$ outputs 1 if $\gamma = \gamma'$ ($\mathcal{A}$ wins the game), else outputs 0.

We define adversary $\mathcal{A}$'s advantage against the Glassbox security of an IBE scheme as:

$$Adv_{\mathcal{A}}^{\mathcal{E},\texttt{GB-ID}} = |Pr(\gamma = \gamma') - \frac{1}{2}| \leq \nu(\kappa)$$

5 Glassbox Attack on Boneh-Franklin IBE

We first briefly look into Boneh-Franklin's construction [5] of a CCA secure IBE (FULLIDENT). Following this, we will demonstrate how an attack can be mounted by a Glassbox adversary despite all secret key involving computations being done in a tamper resistant hardware module.

5.1 Boneh-Franklin CCA Secure IBE (FULLIDENT)

The construction of FULLIDENT achieves CCA security by applying a transformation due to Fujisaki-Okamoto [7] on Boneh-Franklin's CPA secure IBE scheme (BASICIDENT) [5].

- **Setup**(κ): This algorithm is run by the PKG to set up system parameters and MSK. Given an appropriate security parameter κ, the algorithm works as follows:
  - Generate the definition of bilinear map $\langle e, \mathbb{G}_1, \mathbb{G}_2, p \rangle \leftarrow \mathcal{G}(\kappa)$ where both the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are of prime order p and the bilinear map is defined as $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$
  - Pick a random generator $P_1 \in \mathbb{G}_1$
  - Choose the master secret key (MSK) $s \in_R \mathbb{Z}_p^*$
  - Set the master public key $P_{pub} = sP_1$
  - Select cryptographic hash functions:
    - $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$
    - $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^m$ for some $m \in \mathbb{Z}^+$
    - $H_3 : \{0,1\}^m \times \{0,1\}^m \rightarrow \mathbb{Z}_p^*$
    - $H_4 : \{0,1\}^m \rightarrow \{0,1\}^m$

The message space $\mathcal{M} = \{0,1\}^m$ and ciphertext space $\mathcal{C} = \langle \mathbb{G}_1 \times \{0,1\}^m \times \mathbb{G}_1 \times \mathbb{G}_1 \rangle$

- Publish system parameters $\texttt{param} = \langle e, m, \mathbb{G}_1, \mathbb{G}_2, p, P_1, P_{pub}, H_1, H_2, H_3, H_4 \rangle$

– **KeyGen**$(\texttt{ID}, \texttt{param})$: For any arbitrary identity $\texttt{ID} \in \{0,1\}^*$ associated to an user, below are the steps that PKG performs to generate the private key for that identity:
  - Compute $Q_{\texttt{ID}} = H_1(\texttt{ID})$
  - Compute private key $d_{\texttt{ID}} = sQ_{\texttt{ID}}$

– **Encrypt**$(M, \texttt{ID}, \texttt{param})$: The sender executes the algorithm below to encrypt a message $M \in \mathcal{M}$ under public key $\texttt{ID}$:
  - Compute $Q_{\texttt{ID}} = H_1(\texttt{ID})$
  - Choose a random string $\sigma \in_R \{0,1\}^m$
  - Set $r = H_3(\sigma, M)$
  - Compute $g_{\texttt{ID}} = e(Q_{\texttt{ID}}, P_{pub})$
  - Set $C_1 = rP_1$
  - Set $C_2 = \sigma \oplus H_2(g_{\texttt{ID}}^r)$
  - Set $C_3 = M \oplus H_4(\sigma)$
  - Declare the ciphertext to be $C = \langle C_1, C_2, C_3 \rangle$

– **Decrypt**$(C, d_{\texttt{ID}}, \texttt{param})$: The receiver runs this algorithm on a hybrid platform to decrypt the ciphertext $C = \langle C_1, C_2, C_3, C_4 \rangle$ to recover message M. Private key $d_{\texttt{ID}}$ resides in TPM. Computations, as done in RAM/TPM, are marked accordingly.
  - RAM: If $C_1 \notin \mathbb{G}_1^*$, then **ABORT**
  - TPM: Compute $\alpha = e(d_{\texttt{ID}}, C_1)$
  - Send TPM $\xrightarrow{\alpha}$ RAM
  - RAM: Compute $\sigma = C_2 \oplus H_2(\alpha)$
  - RAM: Compute $M = C_3 \oplus H_4(\sigma)$
  - RAM: Compute $r = H_3(\sigma, M)$
  - RAM: If $C_1 \neq rP_1$, then **ABORT**
  - Output M as decryption of C

Remark: During simulation, Glassbox decryption oracle $\mathcal{O}_{\texttt{GB-ID}}$ returns to the adversary the set intermediate values $\mathcal{I} = \{\alpha, H_2(\alpha), \sigma, H_4(\sigma), M, r\}$ which are either computed in RAM or sent to RAM from TPM until abortion/termination of the decryption algorithm. A subtle difference to be observed is that unlike CCA2 decryption oracle which is supposed to divulge M *only* if the consistency check (indicating the correctness of the decrypted message) passes at the final step of the decryption algorithm, the Glassbox decryption oracle always reveals M to the adversary in this particular implementation, no matter decryption is correct or not. Nevertheless, a Glassbox attack does not rely on the correctness of the decrypted message as we will see in section 5.2 and can still be mounted even when a gibberish M is returned. The weakness of the implementation stems from the fact that $\mathcal{O}_{\texttt{GB-ID}}$ returns the same set $\mathcal{I}$ irrespective of whether the decryption succeeds/fails.

5.2 Glassbox Attack on FULLIDENT Scheme

Let the adversary $\mathcal{A}$ play Glassbox security game with the challenger $\mathcal{C}$ as outlined in section 4.5. Following attack can be mounted on FULLIDENT scheme above to exploit Glassbox vulnerability.

- **Phase 1**: $\mathcal{A}$ makes usual key extraction and decryption query to $\mathcal{C}$ as defined in CCA2 security game.
- **Challenge**: $\mathcal{A}$ sends two messages $\langle M_0, M_1\rangle$ and an identity $\mathtt{ID}^*$ to $\mathcal{C}$. $\mathcal{C}$ chooses a random bit $\delta \in_R \{0,1\}$, encrypts M_δ under $\mathtt{ID}^*$ and sends the ciphertext $C^* = \langle C_1^*$, $C_2^*, C_3^* = M_\delta \oplus H_4(\sigma^*)\rangle$ back to $\mathcal{A}$.
- **Phase 2**: $\mathcal{A}$ queries the glassbox oracle $\mathcal{O}_{\mathtt{GB\text{-}ID}}$ for the decryption of the ciphertext $C' = \langle C_1' = C_1^*, C_2' = C_2^*, C_3' \in_R \{0,1\}^m\rangle$ under the challenge identity $\mathtt{ID}^*$ itself. $\mathcal{C}$ hands over intermediate values $\mathcal{I} = \{\alpha', H_2(\alpha'), \sigma', H_4(\sigma'), M', r'\}$ for some arbitrary message M'. Since, $C_1' = C_1^*$ and $C_2' = C_2^*$, evidently $\sigma' = \sigma^*$. $\mathcal{A}$ can trivially recover the challenge message by computing $M_\delta = C_3^* \oplus H_4(\sigma^*)$, thus identifying bit δ always.

6 GBPKE - A Glassbox Resilient PKE Scheme

- **KeyGen**(κ): This algorithm is run by the user to generate public-secret key-pair. Given an appropriate security parameter κ, the algorithm works as follows:
  - Generate the definition of bilinear map $\langle e, \mathbb{G}_1, \mathbb{G}_2, p\rangle \leftarrow \mathcal{G}(\kappa)$ where both the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are of prime order p and the bilinear map is defined as $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$
  - Pick a random generator $P_1 \in \mathbb{G}_1$ and $Q_{\mathtt{ID}}, Y, Z \in_R \mathbb{G}_1$ and random integers $r, s \in_R \mathbb{Z}_p$
  - Compute $P_{pub} = sP_1$
  - Select cryptographic hash functions:
    - $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^m$ for some $m \in \mathbb{Z}^+$
    - $H_3 : \mathbb{G}_1 \times \{0,1\}^m \times \mathbb{G}_1 \rightarrow \mathbb{Z}_p$
    - $H_4 : \mathbb{G}_2 \times \mathbb{G}_1 \rightarrow \mathbb{Z}_p$

    The message space $\mathcal{M} = \{0,1\}^m$ and ciphertext space $\mathcal{C} = \langle \mathbb{G}_1 \times \{0,1\}^m \times \mathbb{G}_1 \times \mathbb{G}_1\rangle$
  - Compute secret key $d_{\mathtt{ID}} = \langle rsQ_{\mathtt{ID}}, r^{-1}\rangle = \langle d_{\mathtt{ID1}}, d_{\mathtt{ID2}}\rangle$
  - Publish public key $\mathtt{pubkey} = \langle e, m, \mathbb{G}_1, \mathbb{G}_2, p, P_1, P_{pub}, H_2, H_3, H_4, Q_{\mathtt{ID}}, Y, Z\rangle$
- **Encrypt**(M, $\mathtt{pubkey}$): The sender executes the algorithm below to encrypt a message $M \in \mathcal{M}$
  - Choose random $u \in_R \mathbb{Z}_p$ and $X \in_R \mathbb{G}_1$
  - Set $C_1 = uP_1$
  - Compute $g_{\mathtt{ID}} = e(Q_{\mathtt{ID}}, P_{pub})$
  - Set $C_2 = M \oplus H_2(g_{\mathtt{ID}}^u)$
  - Compute $t = H_3(C_1, C_2, Q_{\mathtt{ID}})$
  - Compute $h = H_4(e(u(tP_1 + X), P_1), Q_{\mathtt{ID}})$
  - Set $C_3 = u(hY + Z)$

- Set $C_4 = X$
- Declare the ciphertext to be $C = \langle C_1, C_2, C_3, C_4 \rangle$

– **Decrypt**$(C, d_{\text{ID}}, \texttt{pubkey})$: The receiver runs this algorithm on a hybrid platform to decrypt the ciphertext $C = \langle C_1, C_2, C_3, C_4 \rangle$ to recover message M. Private key component d_{ID1} resides in RAM while d_{ID2} resides in TPM. Computations, as done in RAM/TPM, are marked accordingly.
  - RAM: Compute $t = H_3(C_1, C_2, Q_{\text{ID}})$
  - RAM: Compute $h = H_4(e(tP_1 + C_4, C_1), Q_{\text{ID}})$
  - RAM: If $e(C_3, P_1) \overset{?}{=} e(hY + Z, C_1)$, then proceed as follows:
    - ∗ RAM: Compute $\alpha = e(d_{\text{ID1}}, C_1)$
    - ∗ Send RAM $\xrightarrow{\alpha}$ TPM
    - ∗ TPM: $\beta = (\alpha)^{d_{\text{ID2}}}$
    - ∗ Send TPM $\xrightarrow{\beta}$ RAM
    - ∗ RAM: Recover $M = C_2 \oplus H_2(\beta)$

    Else **ABORT**

Remark: During simulation, Glassbox decryption oracle $\mathcal{O}_{\texttt{GB-ID}}$ returns to the adversary all such intermediate values which are either computed in RAM or sent to RAM from TPM until abortion/termination of the decryption algorithm. The oracle initially hands over a set of values $\mathcal{I}_{failure} = \{t, e(tP_1 + C_4, C_1), h, e(C_3, P_1), e(hY + Z, C_1)\}$. Further, if the ciphertext integrity check $(e(C_3, P_1) \overset{?}{=} e(hY + Z, C_1))$ succeeds, the decryption algorithm proceeds to compute an additional set of values $\mathcal{I}_{consistent} = \{\alpha, \beta, H_2(\beta), M\}$, otherwise **ABORT**s. Essentially, $\mathcal{O}_{\texttt{GB-ID}}$ leaks the set of values $\mathcal{I}_{success} = \{\mathcal{I}_{failure} \cup \mathcal{I}_{consistent}\}$ for a successful run of the algorithm.

Protocol Correctness: For honest execution of protocols, 'correctness' will hold when the following conditions are satisfied:

1. The consistency check passes for a well-formed ciphertext $C = \langle C_1, C_2, C_3, C_4 \rangle$: $e(C_3, P_1) = e(hY + Z, C_1)$
2. $g_{\texttt{ID}}^u = e(d_{\texttt{ID1}}, C_1)^{d_{\text{ID2}}}$

1. **Proof of Assertion 1**: For a well-formed ciphertext, $C_1 = uP_1, \exists u \in \mathbb{Z}_p$. The same u is reused in constructing $C_3 = u(hY + Z)$. Due to bilinearity property of the map e,
$$e(C_3, P_1) = e(u(hY + Z), P_1) = e(hY + Z, uP_1) = e(hY + Z, C_1)$$
2. **Proof of Assertion 2**: Recall that, secret key $d_{\texttt{ID1}} = rsP_1$ and $d_{\texttt{ID2}} = r^{-1}$. $P_{pub} = sP_1, C_1 = uP_1$. Due to bilinearity property of the map e,

$$\begin{aligned} g_{\texttt{ID}}^u = e(Q_{\text{ID}}, P_{pub})^u = e(Q_{\text{ID}}, sP_1)^u = e(sQ_{\text{ID}}, uP_1) \\ = [e(sQ_{\text{ID}}, C_1)^r]^{r^{-1}} = e(rsQ_{\text{ID}}, C_1)^{r^{-1}} = e(d_{\texttt{ID1}}, C_1)^{d_{\text{ID2}}} \end{aligned}$$

Theorem 1: *Suppose hash functions H_2, H_3, H_4 are treated as random oracles. Consider an IND-GB adversary $\mathcal{B}$ having an advantage $\nu(\kappa)$ against the scheme GBPKE. Let $\mathcal{B}$ make at most $q_{H_2} > 0$ hash queries to its challenger $\mathcal{C}$. Then $\mathcal{C}$ can solve CBDH problem in the groups generated by $\mathcal{G}$ with at least an advantage $2\nu(\kappa)/q_{H_2}$.*

Proof: Detailed proof of the theorem above is presented in Appendix A.

7 GBIBE - A Glassbox Resilient IBE Scheme

- **Setup**(κ): This algorithm is run by the PKG to set up system parameters and MSK. Given an appropriate security parameter κ, the algorithm woks as follows:
  - Generate the definition of bilinear map $\langle e, \mathbb{G}_1, \mathbb{G}_2, p\rangle \leftarrow \mathcal{G}(\kappa)$ where both the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are of prime order p and the bilinear map is defined as $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$
  - Pick a random generator $P_1 \in \mathbb{G}_1$ and $Y, Z \in_R \mathbb{G}_1$
  - Choose the master secret key (MSK) $s \in_R \mathbb{Z}_p$
  - Set the master public key $P_{pub} = sP_1$
  - Select cryptographic hash functions:
    * $H_1 : \{0,1\}^* \to \mathbb{G}_1$
    * $H_2 : \mathbb{G}_2 \to \{0,1\}^m$ for some $m \in \mathbb{Z}^+$
    * $H_3 : \mathbb{G}_1 \times \{0,1\}^m \times \mathbb{G}_1 \to \mathbb{Z}_p$
    * $H_4 : \mathbb{G}_2 \times \mathbb{G}_1 \to \mathbb{Z}_p$

    The message space $\mathcal{M} = \{0,1\}^m$ and ciphertext space $\mathcal{C} = \langle \mathbb{G}_1 \times \{0,1\}^m \times \mathbb{G}_1 \times \mathbb{G}_1\rangle$
  - Publish system parameters $\texttt{param} = \langle e, m, \mathbb{G}_1, \mathbb{G}_2, p, P_1, P_{pub}, H_1, H_2, H_3, H_4, Y, Z\rangle$
- **KeyGen**$(\texttt{ID}, \texttt{param})$: For any arbitrary identity $\texttt{ID} \in \{0,1\}^*$ associated to an user, below are the steps that PKG performs to generate the private key for that identity:
  - Compute $Q_{\texttt{ID}} = H_1(\texttt{ID})$
  - Choose a random $r \in_R \mathbb{Z}_p$
  - Compute private key $d_{\texttt{ID}} = \langle rsQ_{\texttt{ID}}, r^{-1}\rangle = \langle d_{\texttt{ID1}}, d_{\texttt{ID2}}\rangle$
- **Encrypt**$(M, \texttt{ID}, \texttt{param})$: The sender executes the algorithm below to encrypt a message $M \in \mathcal{M}$ under public key ID:
  - Choose random $u \in_R \mathbb{Z}_p$ and $X \in_R \mathbb{G}_1$
  - Set $C_1 = uP_1$
  - Compute $Q_{\texttt{ID}} = H_1(\texttt{ID})$
  - Compute $g_{\texttt{ID}} = e(Q_{\texttt{ID}}, P_{pub})$
  - Set $C_2 = M \oplus H_2(g_{\texttt{ID}}^u)$
  - Compute $t = H_3(C_1, C_2, Q_{\texttt{ID}})$
  - Compute $h = H_4(e(u(tP_1 + X), P_1), Q_{\texttt{ID}})$
  - Set $C_3 = u(hY + Z)$
  - Set $C_4 = X$
  - Declare the ciphertext to be $C = \langle C_1, C_2, C_3, C_4\rangle$

– **Decrypt**$(C, d_{\texttt{ID}}, \texttt{param})$: The receiver runs this algorithm on a hybrid platform to decrypt the ciphertext $C = \langle C_1, C_2, C_3, C_4 \rangle$ to recover message M. Private key component $d_{\texttt{ID1}}$ resides in RAM while $d_{\texttt{ID2}}$ resides in TPM. Computations, as done in RAM/TPM, are marked accordingly.
  - RAM: Compute $Q_{\texttt{ID}} = H_1(\texttt{ID})$
  - RAM: Compute $t = H_3(C_1, C_2, Q_{\texttt{ID}})$
  - RAM: Compute $h = H_4(e(tP_1 + C_4, C_1), Q_{\texttt{ID}})$
  - RAM: If $e(C_3, P_1) \overset{?}{=} e(hY + Z, C_1)$, then proceed as follows:
    - ∗ RAM: Compute $\alpha = e(d_{\texttt{ID1}}, C_1)$
    - ∗ Send $\mathsf{RAM} \xrightarrow{\alpha} \mathsf{TPM}$
    - ∗ TPM: $\beta = (\alpha)^{d_{\texttt{ID2}}}$
    - ∗ Send $\mathsf{TPM} \xrightarrow{\beta} \mathsf{RAM}$
    - ∗ RAM: Recover $M = C_2 \oplus H_2(\beta)$

    Else **ABORT**

Remark: During simulation, Glassbox decryption oracle $\mathcal{O}_{\texttt{GB-ID}}$ returns to the adversary all such intermediate values which are either computed in RAM or sent to RAM from TPM until abortion/termination of the decryption algorithm. The oracle initially hands over a set of values $\mathcal{I}_{failure} = \{Q_{\texttt{ID}}, t, e(tP_1 + C_4, C_1), h, e(C_3, P_1), e(hY + Z, C_1)\}$. Further, if the ciphertext integrity check $(e(C_3, P_1) \overset{?}{=} e(hY + Z, C_1))$ succeeds, the decryption algorithm proceeds to compute an additional set of values $\mathcal{I}_{consistent} = \{\alpha, \beta, H_2(\beta), M\}$, otherwise **ABORT**s. Essentially, $\mathcal{O}_{\texttt{GB-ID}}$ leaks the set of values $\mathcal{I}_{success} = \{\mathcal{I}_{failure} \cup \mathcal{I}_{consistent}\}$ for a successful run of the algorithm.

Protocol Correctness: Directly follows from the protocol correctness of GBPKE as shown earlier.

Theorem 2: *Suppose hash functions H_1, H_2, H_3, H_4 are treated as random oracles. Consider an IND-ID-GB adversary $\mathcal{A}$ having an advantage $\nu(\kappa)$ against the scheme GBIBE. Let $\mathcal{A}$ make at most $q_X > 0$ secret key extraction queries to its challenger $\mathcal{B}$. Then $\mathcal{B}$ can be viewed as an IND-GB adversary having at least an advantage $\nu(\kappa)/\hat{e}(1 + q_X)$ against the scheme GBPKE in the security game played with $\mathcal{C}$. Here, $\hat{e} \approx 2.71$ is the base of natural logarithm.*

Proof: Detailed proof of the theorem above is presented in Appendix B.

8 Conclusion

In this paper, we have defined the Glassbox security model for identity based cryptosystems. The notion introduced here is stronger than *de-facto* standard CCA2 formalism. Thus several CCA2 secure protocols can be shown vulnerable to the attack by a Glassbox adversary. This new security paradigm closely captures *real-world* threats as posed by MEMORY SCRAPER type malwares. We have proposed both PKE as well as IBE protocols withstanding such attacks. Implementations of the same are intended to be deployed on hybrid systems equipped with *even* a minimal tamper resistant hardware module capable of performing group exponentiation operation only. Such systems may be of immense interest to thwart, mitigate or prevent state-of-the-art malware attacks.

9 Open Directions

Our IBE scheme is provably secure in random oracle model. The construction requires the challenger to inject one parameter of **CBDH** instance inside $Q_{\mathtt{ID}}$ which the challenger needs to have control on. That was the bottleneck preventing us from proving the scheme secure in standard model. Designing an IBE protocol secure in standard model capable of withstanding Ram Scraper like malware attack is an interesting problem which we leave open for further research.

References

1. Tinypbc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. Computer Communications 34(3), 485 – 493 (2011)
2. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Theory of Cryptography. LNCS, vol. 5444, pp. 474–495. Springer Berlin Heidelberg (2009)
3. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Advances in Cryptology - CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer Berlin Heidelberg (2009)
4. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Advances in Cryptology - CRYPTO '98. LNCS, vol. 1462, pp. 26–45. Springer Berlin Heidelberg (1998)
5. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Advances in Cryptology - CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer Berlin Heidelberg (2001)
6. Coron, J.S.: On the exact security of full domain hash. In: Advances in Cryptology CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer Berlin Heidelberg (2000)
7. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of Cryptology 26(1), 80–101 (2013)
8. Huq, N.: A trend micro research paper, pos ram scraper malware - past, present, and future (2014), `http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-pos-ram-scraper-malware.pdf`
9. Inc., V.: Visa data security alert, debugging software memory parsing vulnerability (2008), `http://usa.visa.com/download/merchants/debugging_software_memory.pdf`
10. Juma, A., Vahlis, Y.: Protecting cryptographic keys against continual leakage. In: Advances in Cryptology - CRYPTO 2010. LNCS, vol. 6223, pp. 41–58. Springer Berlin Heidelberg (2010)
11. Kiltz, E., Pietrzak, K.: Leakage resilient elgamal encryption. In: Advances in Cryptology - ASIACRYPT 2010. LNCS, vol. 6477, pp. 595–612. Springer Berlin Heidelberg (2010)
12. Microsoft:
13. Rackoff, C., Simon, D.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Advances in Cryptology - CRYPTO '91. LNCS, vol. 576, pp. 433–444. Springer Berlin Heidelberg (1992)
14. Sarr, A., Elbaz-Vincent, P., Bajard, J.C.: A new security model for authenticated key agreement. In: Security and Cryptography for Networks. LNCS, vol. 6280, pp. 219–234. Springer Berlin Heidelberg (2010)
15. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G., Chaum, D. (eds.) Advances in Cryptology, LNCS, vol. 196, pp. 47–53. Springer Berlin Heidelberg (1985)

16. (TCG), T.C.G.: Trusted platform module library, part i - architecture, family 2.0, level 00, revision 01.16 (2014), `http://www.trustedcomputinggroup.org/files/static_page_files/8C56AE3E-1A4B-B294-D0F43097156A55D8/TPMRev2.0Part1-Architecture01.16.pdf`
17. Vivek, S.S., Selvi, S.S.D., Rangan, C.P.: Stronger public key encryption schemes withstanding ram scraper like attacks. Cryptology ePrint Archive, Report (2012)
18. Yarom, Y., Falkner, K.: Flush+reload: A high resolution, low noise, l3 cache side-channel attack. In: Proceedings of the 23rd USENIX Conference on Security Symposium. pp. 719–732. SEC'14, USENIX Association (2014)

10 Appendix A: Security of GBPKE

Theorem 1: *Suppose hash functions H_2, H_3, H_4 are treated as random oracles. Consider an IND-GB adversary $\mathcal{B}$ having an advantage $\nu(\kappa)$ against the scheme GBPKE. Let $\mathcal{B}$ make at most $q_{H_2} > 0$ hash queries to its challenger $\mathcal{C}$. Then $\mathcal{C}$ can solve CBDH problem in the groups generated by $\mathcal{G}$ with at least an advantage $2\nu(\kappa)/q_{H_2}$.*

Proof: Let $\mathcal{B}$ be an IND-GB adversary against GBPKE. We can construct a challenger $\mathcal{C}$ that interacts with $\mathcal{B}$ to solve the **CBDH** problem with negligible advantage.

Challenger $\mathcal{C}$ is given as input the following **CBDH** parameters $\langle p, G_1, G_2, e\rangle$ and a random instance $\langle P_1, aP_1, bP_1, cP_1\rangle$ of the **CBDH** problem where $\langle a, b, c\rangle \in_R \mathbb{Z}_p^3$ and $G_1 = \langle P_1\rangle$. Let $D = e(P_1, P_1)^{abc}$ be the solution to the CBDH problem which $\mathcal{C}$ is attempting to find by interacting with $\mathcal{B}$ as follows:

– **Setup**: $\mathcal{C}$ generates `pubkey` $= \langle e, m, \mathbb{G}_1, \mathbb{G}_2, p, P_1, P_{pub}, H_2, H_3, H_4, Q_{\texttt{ID}}, Y, Z\rangle$ by setting $P_{pub} = aP_1$ and $Q_{\texttt{ID}} = bP_1$. It picks $(r, t', z', y') \in_R \mathbb{Z}_p^4$. The `secret key` $d_{\texttt{ID}} = \langle raQ_{\texttt{ID}}, r^{-1}\rangle$ where $\langle aQ_{\texttt{ID}} = abP_1\rangle$ is unknown to $\mathcal{C}$. Our proof views hash functions $\langle H_2, H_3, H_4\rangle$ as random oracles $\mathcal{O}_{H2}, \mathcal{O}_{H3}, \mathcal{O}_{H4}$ controlled by $\mathcal{C}$ as described below:

$\boldsymbol{\mathcal{O}_{H2}}$: Let $\mathcal{B}$ ask $\mathcal{O}_{H2}$ for the value of $H_2(D_i), D_i \in \mathbb{G}_2$. $\mathcal{C}$ maintains a list $\mathcal{L}_{H2}$ of tuples of the form $\langle D_i, h_{2i}\rangle$.

- If the tuple $\langle D_i, h_{2i}\rangle$ is present in $\mathcal{L}_{H2}$, it returns $H_2(D_i) = h_{2i}$.
- Otherwise, $\mathcal{C}$ picks a random string $h_{2i} \in \{0,1\}^m$, add $\langle D_i, h_{2i}\rangle$ to $\mathcal{L}_{H2}$ and responds with $H_2(D_i) = h_{2i}$ to $\mathcal{B}$.

$\boldsymbol{\mathcal{O}_{H3}}$: Let $\mathcal{B}$ ask $\mathcal{O}_{H3}$ for the value of $H_3(C_{1i}, C_{2i}, Q_i)$. $\mathcal{C}$ maintains a list $\mathcal{L}_{H3}$ of tuples of the form $\langle C_{1i}, C_{2i}, Q_i, t_i\rangle$.

- If the tuple $\langle C_{i1}, C_{i2}, Q_i, t_i\rangle$ is present in $\mathcal{L}_{H3}$, it returns $H_3(C_{i1}, C_{i2}, Q_i) = t_i$
- Otherwise, $\mathcal{C}$ picks $t_i \in_R \mathbb{Z}_p$, adds $\langle C_{1i}, C_{2i}, Q_i, t_i\rangle$ to $\mathcal{L}_{H3}$ and responds with $H_3(C_{1i}, C_{2i}, Q_i) = t_i$ to $\mathcal{B}$.

$\boldsymbol{\mathcal{O}_{H4}}$: Let $\mathcal{B}$ ask $\mathcal{O}_{H4}$ for the value of $H_4(X_i, Q_i)$, $X_i \in \mathbb{G}_2$. $\mathcal{C}$ maintains a list $\mathcal{L}_{H4}$ of tuples of the form $\langle X_i, Q_i, h_i \rangle$.

- If the tuple $\langle X_i, Q_i, h_i \rangle$ is present in $\mathcal{L}_{H4}$, it returns $H_4(X_i, Q_i) = h_i$,
- Otherwise, $\mathcal{C}$ picks $h_i \in_R \mathbb{Z}_p$, adds $\langle X_i, Q_i, h_i \rangle$ to $\mathcal{L}_{H4}$ and responds with $H_2(X_i, Q_i) = h_i$ to $\mathcal{B}$.

$\mathcal{C}$ sets $h^* = H_4(e(t'P_1, cP_1), Q_{\texttt{ID}})$, $Y = \frac{1}{h^*}(Q_{\texttt{ID}} + y'P_1)$, $Z = (-Q_{\texttt{ID}} + z'P_1)$

– **Phase 1**: $\mathcal{B}$ makes $\mathcal{O}_{\texttt{GB-PK}}$ queries to $\mathcal{C}$. Let $\mathcal{B}$ request decryption of $C = \langle C_1, C_2, C_3, C_4 \rangle$ to $\mathcal{C}$. $\mathcal{C}$ answers the queries as follows:
  - Computes $t = H_3(C_1, C_2, Q_{\texttt{ID}})$
  - Computes $h = H_4(e(tP_1 + C_4, C_1), Q_{\texttt{ID}})$
  - If $e(C_3, P_1) \overset{?}{=} e(hY + Z, C_1)$, then proceed as follows:
    - $*$ So far $\mathcal{C}$ could compute all the values as it knew the pre-requisite ones. To compute $\alpha = e(d_{\texttt{ID1}}, C_1)$, knowledge of $d_{\texttt{ID1}}$ is necessary. Simulation environment implicitly sets $d_{\texttt{ID1}} = rabP_1$, which $\mathcal{C}$ is not aware of. Hence, $\mathcal{C}$ simulates α as follows:

$$\begin{aligned} \alpha &= e(d_{\texttt{ID1}}, C_1) \\ &= e(rabP_1, uP_1) \\ &= e(u(bP_1), aP_1)^r \\ &= e(uQ_{\texttt{ID}}, P_{pub})^r \end{aligned}$$

$$\begin{aligned} C_3 &= u(hY + Z) \\ &= u\left(\frac{h}{h^*}(Q_{\texttt{ID}} + y'P_1) - Q_{\texttt{ID}} + z'P_1\right) \\ &= u\left(\left(\frac{h}{h^*} - 1\right) Q_{\texttt{ID}} + \frac{h}{h^*}y'P_1 + z'P_1\right) \\ &= \left(\frac{h}{h^*} - 1\right) uQ_{\texttt{ID}} + \frac{h}{h^*}y'C_1 + z'C_1 \\ \therefore uQ_{\texttt{ID}} &= \left(\frac{h}{h^*} - 1\right)^{-1} \left(C_3 - \frac{h}{h^*}y'C_1 - z'C_1\right) \end{aligned}$$

    - $*$ Computes $\beta = \alpha^{d_{\texttt{ID2}}}$
    - $*$ Computes $h_2 = H_2(\beta)$
    - $*$ Computes $M = C_2 \oplus h_2$
    - $*$ $\mathcal{C}$ returns $\mathcal{I} = \langle t, e(tP_1 + C_4), h, e(C_3, P_1), e(hY + Z), \alpha, \beta, H_2(\beta), M \rangle$
  - Else **ABORT**s and returns $\mathcal{I} = \langle t, e(tP_1 + C_4), h, e(C_3, P_1), e(hY + Z), \_, \_, \_, \_ \rangle$

– **Challenge**: After $\mathcal{B}$ decides **Phase 1** training is over, $\mathcal{B}$ gives two messages of equal length (M_0, M_1) which it wishes to be challenged on. Recall that y', z', t' were chosen during setup phase. $\mathcal{C}$ generates the challenge ciphertext $C^* = \langle C_1^*, C_2^*, C_3^*, C_4^* \rangle$ as follows:
  - Sets $C_1^* = cP_1$, where cP_1 is one of the input to the CBDH problem.
  - Choose $C_2^* \in_R \{0, 1\}^m$

- Computes $C_3^* = (y' + z')C_1^*$
- Compute $t = H_3(C_1^*, C_2^*, Q_{\text{ID}})$
- Calculates $C_4^* = (t' - t)P_1$

$\mathcal{C}$ sends this challenge ciphertext C^* to $\mathcal{B}$.

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

Proof: Setting $C_1^* = cP_1$ in the simulation implicitly sets $u = c$. We will prove our claim in two steps. Firstly, we will show that $h^* = H_4(e(tP_1 + C_4^*, C_1^*), Q_{\text{ID}})$ and secondly $C_3^* = (y' + z')C_1^*$ is of the form $c(h^*Y + Z)$

$$\begin{aligned}
& H_4(e(tP_1 + C_4^*, C_1^*), Q_{\text{ID}}) \\
&= H_4(e(tP_1 + (t' - t)P_1, C_1^*), Q_{\text{ID}}) \\
&= H_4(e(t'P_1, cP_1), Q_{\text{ID}}) \\
&= h^*
\end{aligned}
\qquad\qquad
\begin{aligned}
& c(h^*Y + Z) \\
&= c(Q_{\text{ID}} + y'P_1 - Q_{\text{ID}} + z'P_1)) \\
&= c(y'P_1 + z'P_1) \\
&= y'(cP_1) + z'(cP_1) \\
&= (y' + z')C_1^* \\
&= C_3^*
\end{aligned}$$

– **Phase 2**: This phase is similar to **Phase 1** with the constraint that $\mathcal{B}$ should not make any decryption query on C^*

– **Guess**: Eventually, $\mathcal{B}$ outputs its guess $\gamma' \in \{0, 1\}$ for γ. $\mathcal{C}$ picks a random tuple $\langle D_i, h_{2i} \rangle$ from $\mathcal{L}_{H2}$ list and outputs D_i as a solution to the **CBDH** problem.

By definition (as claimed in **Theorem 1**) of adversary $\mathcal{B}$, its advantage in IND-GB game is

$$Adv_{\mathcal{B}} = |\Pr[\gamma = \gamma'] - \frac{1}{2}| \geq \nu(\kappa) \tag{1}$$

Recall that $D = g_{\text{ID}}^u$ is the solution to **CBDH** instance embedded in the game. Suppose, E be the event of adversary $\mathcal{B}$ asking for the value of $h_{2D} = H_2(D) \in_R \{0, 1\}^m$ to $\mathcal{O}_{H2}$ at some point during the lifetime of the game. Because, h_{2D} is uniformly distributed in the view of the adversary $\mathcal{B}$ and is bitwise XOR*ed* with the message m, it behaves identical to one-time-pad (OTP). In case $\mathcal{B}$ does not issue a query for $H_2(D)$, it gets no advantage more than random guessing in distinguishing the two possible decryption of challenge ciphertext:

$$|\Pr[\gamma = \gamma' \mid \overline{E}] = \frac{1}{2} \tag{2}$$

In case $\mathcal{B}$ issues a query for $H_2(D)$, it can certainly distinguish between the two possible decryption of challenge ciphertext:

$$|\Pr[\gamma = \gamma' \mid E] = 1 \tag{3}$$

Now,

$$\begin{aligned}
\Pr[\gamma = \gamma'] &= \Pr[\gamma = \gamma' \mid E]\Pr[E] + \Pr[\gamma = \gamma' \mid \overline{E}]\Pr[\overline{E}] \\
&\leq \Pr[E] + \frac{1}{2}\Pr[\overline{E}] = \Pr[E] + \frac{1}{2}\Pr(1 - \Pr[E]) = \frac{1}{2} + \frac{1}{2}\Pr[E]
\end{aligned} \tag{4}$$

Again,

$$\Pr[\gamma = \gamma'] \geq \Pr[\gamma = \gamma' \mid \overline{E}]\Pr[\overline{E}] = \frac{1}{2}(1 - \Pr[E]) = \frac{1}{2} - \frac{1}{2}\Pr[E] \quad (5)$$

Combining equations [1], [4] and [5], $\frac{1}{2}\Pr[E] \geq |\Pr[\gamma = \gamma'] - \frac{1}{2}| \geq \nu(\kappa) \Longrightarrow \Pr[E] \geq \nu(\kappa)$. At the end of the simulation, D appears in $\mathcal{L}_{H2}$ with probability at least $\nu(\kappa)$. $\mathcal{B}$ arbitrarily picks a value from the list $\mathcal{L}_{H2}$ and returns it as solution to **CBDH** instance. Thus, the probability of hitting the correct solution is at least $2\nu(\kappa)/q_{H2}$.

11 Appendix B: Security of GBIBE

Theorem 2: *Suppose hash functions H_1, H_2, H_3, H_4 are treated as random oracles. Consider an IND-ID-GB adversary $\mathcal{A}$ having an advantage $\nu(\kappa)$ against the scheme GBIBE. Let $\mathcal{A}$ make at most $q_X > 0$ secret key extraction queries to its challenger $\mathcal{B}$. Then $\mathcal{B}$ can be viewed as an IND-GB adversary having at least an advantage $\nu(\kappa)/\hat{e}(1+q_X)$ against the scheme GBPKE in the security game played with $\mathcal{C}$. Here, $\hat{e} \approx 2.71$ is the base of natural logarithm.*

Proof: We will first convert a IND-ID-GB attack on GBIBE to a IND-GB attack on GBPKE. This reduction, combined with **Theorem 1**, proves the claim above.

Let $\mathcal{A}$ be an IND-ID-GB adversary against GBIBE. We can construct an adversary $\mathcal{B}$ that interacts with $\mathcal{A}$ to solve the challenge by challenger $\mathcal{C}$ in IND-GB security game. Note that, $\mathcal{B}$ plays the role of a challenger in the former security game while an adversary in the later.

$$\mathcal{C} \xrightarrow[\mathcal{B}\text{ is Adversary}]{\text{IND-GB}} \mathcal{B} \xrightarrow[\mathcal{B}\text{ is Challenger}]{\text{IND-ID-GB}} \mathcal{A}$$

$\mathcal{C}$ executes `KeyGen` of GBPKE to generate public key $\texttt{pubkey} = \langle e, m, \mathbb{G}_1, \mathbb{G}_2, p, P_1, P_{pub}, H_2, H_3, H_4, Q_{\text{ID}}, Y, Z\rangle$ and secret key $d_{\texttt{ID}} = \langle rsQ_{\texttt{ID}}, r^{-1}\rangle$. $\mathcal{C}$ gives `pubkey` to adversary $\mathcal{B}$. In the challenge phase of IND-GB game, $\mathcal{B}$ will supply two messages M_0 and M_1 to $\mathcal{C}$ and receive the encryption of M_b under `pubkey` where $b \in \{0,1\}$. After receiving polynomial amount of training by querying secret key extraction and Glassbox decryption oracles, $\mathcal{B}$'s task is to emit its guess $b' \in \{0,1\}$ for b.

$\mathcal{B}$ simulates the IND-ID-GB game environment for $\mathcal{A}$ as follows:

- **Setup**: $\mathcal{B}$ announces GBIBE system parameters $\texttt{param} = \langle e, m, \mathbb{G}_1, \mathbb{G}_2, p, P_1, P_{pub}, H_1, H_2, H_3, H_4, Y, Z\rangle = \langle \texttt{pubkey} - Q_{\texttt{ID}} + H_1\rangle$. Our proof views hash function H_1 as a random oracle $\mathcal{O}_{H1}$ controlled by $\mathcal{B}$.

$\boldsymbol{\mathcal{O}_{H1}}$: Let $\mathcal{A}$ ask $\mathcal{O}_{H1}$ for the value of $H_1(\texttt{ID}_i)$. $\mathcal{B}$ maintains a list $\mathcal{L}_{H1}$ of tuples of the form $\langle \texttt{ID}_j, Q_j, v_j, c_j, r_j\rangle$.
- If $\texttt{ID}_i$ is already queried, corresponding tuple $\langle \texttt{ID}_i, Q_i, v_i, c_i, r_i\rangle$ is returned where $H_1(\texttt{ID}_i) = Q_i$.
- Otherwise, $\mathcal{B}$ selects $v_i, r_i \in \mathbb{Z}_p$. $\mathcal{B}$ tosses a random `coin` $\in \{0,1\}$ with $Pr[coin = 0] = \lambda$. Calculation of exact value of λ requires some trade-off to

be considered and therefore will be computed later. We will use a technique similar to Coron [6]. Queries will be answered differently depending on the outcome of the toss of a $\texttt{coin}$ with bias δ. If $\texttt{coin} = 0$, $Q_i = v_i P_1 \in \mathbb{G}_1$, else $Q_i = v_i Q_{\texttt{ID}} \in \mathbb{G}_1$. $\mathcal{B}$ sets $H_1(\texttt{ID}_i) = Q_i$. $\langle \texttt{ID}_i, Q_i, v_i, \texttt{coin}_i, r_i \rangle$ is added to $\mathcal{L}_{H1}$ and the same is returned as response.

$\boldsymbol{\mathcal{O}_{H2}}$: Let $\mathcal{A}$ ask $\mathcal{O}_{H2}$ for the value of $H_2(g_i^{u_i})$. $\mathcal{B}$ forwards the query unmodified to $\mathcal{C}$ and the response from $\mathcal{C}$ is returned back to $\mathcal{A}$.

$\boldsymbol{\mathcal{O}_{H3}}$: Let $\mathcal{A}$ ask $\mathcal{O}_{H3}$ for the value of $H_3(C_{1i}, C_{2i}, Q_i)$. $\mathcal{B}$ scans the list $\mathcal{L}_{H1}$ for an entry of the form $\langle \texttt{ID}_i, Q_i, v_i, \texttt{coin}_i, r_i \rangle$. If such a tuple is found, $\mathcal{B}$ rewrites the query as $\langle v_i C_{1i}, C_{2i}, Q_i \rangle$, otherwise $\mathcal{B}$ retains the query *as-is*. $\mathcal{B}$ forwards the query to $\mathcal{C}$ and the response from $\mathcal{C}$ is returned back to $\mathcal{A}$. Note that, rewriting the query does not alter $\mathcal{A}$'s view of the interaction; as the output from H_3 is still random and parameterized by C_{1i} while $< C_{1i}, v_i C_{1i} >$ being correlated.

$\boldsymbol{\mathcal{O}_{H4}}$: Let $\mathcal{A}$ ask $\mathcal{O}_{H4}$ for the value of $H_4(\tau, Q_i), \tau \in \mathbb{G}_2$. $\mathcal{B}$ scans the list $\mathcal{L}_{H1}$ for an entry of the form $\langle \texttt{ID}_i, Q_i, v_i, \texttt{coin}_i, r_i \rangle$. If such a tuple is found, $\mathcal{B}$ rewrites the query as $\langle \tau^{v_i}, Q_i \rangle$, otherwise $\mathcal{B}$ retains the query *as-is*. $\mathcal{B}$ forwards the query to $\mathcal{C}$ and the response from $\mathcal{C}$ is returned back to $\mathcal{A}$.

– **Phase 1 – Extraction Query**: Suppose, $\mathcal{A}$ issues secret key extraction query for identity $\texttt{ID}_i$. $\mathcal{B}$ invokes $\mathcal{O}_{H1}$ to obtain the tuple $\langle \texttt{ID}_i, Q_i, v_i, \texttt{coin}_i, r_i \rangle$ where $Q_i = H_1(\texttt{ID}_i)$.
  - If $\texttt{coin}_i = 0$, $\mathcal{B}$ knows that $Q_i = v_i P_1$. It computes the secret key $d_{i1} = r_i s(v_i P_1) = r_i v_i (s P_1) = r_i v_i P_{pub}$ and $d_{i2} = r_i^{-1}$. $\mathcal{B}$ returns $d_i = \langle d_{i1}, d_{i2} \rangle$ to the attacker.
  - If $\texttt{coin}_i = 1$, $\mathcal{B}$ terminates. Attack on GBIBE fails prematurely.

– **Phase 1 – Glassbox Decryption Query**: Let $\mathcal{A}$ ask for the decryption of ciphertext $C_i = \langle C_{i1}, C_{i2}, C_{i3}, C_{i4} \rangle$ under identity $\texttt{ID}_i$. $\mathcal{B}$ invokes $\mathcal{O}_{H1}$ to obtain the tuple $\langle \texttt{ID}_i, Q_i, v_i, \texttt{coin}_i, r_i \rangle$ where $Q_i = H_1(\texttt{ID}_i)$.
  - If $\texttt{coin}_i = 0$, $\mathcal{B}$ runs the key extraction query itself to retrieve the secret key d_i of $\texttt{ID}_i$. It uses d_i to respond to the query and returns intermediate values $\mathcal{I}$ along with the message, if correctly decrypted.
  - If $\texttt{coin}_i = 1$, $\mathcal{B}$ has no direct means to know the secret key of $\texttt{ID}_i$. So, $\mathcal{B}$ queries its IND-GB challenger $\mathcal{C}$ for the decryption of $C_i' = \langle v_i C_{i1}, C_{i2}, v_i C_{i3}, C_{i4} \rangle$ and returns intermediate values $\mathcal{I}$ along with the message, if correctly decrypted.

– **Challenge**: After adequate training, $\mathcal{A}$ emits a message pair $\{M_0, M_1\}$ and an identity $\texttt{ID}^*$ to be challenged on. $\mathcal{B}$ invokes $\mathcal{O}_{H1}$ to obtain the tuple $\langle \texttt{ID}^*, Q^*, v^*, \texttt{coin}^*, r^* \rangle$ where $Q^* = H_1(\texttt{ID}^*)$.

- If $\texttt{coin}^* = 0$, $\mathcal{B}$ terminates. Attack on GBIBE fails prematurely.
- If $\texttt{coin}^* = 1$, $\mathcal{B}$ forwards the message pair $\{M_0, M_1\}$ wishing to be challenged on to its challenger $\mathcal{C}$ in $\mathsf{IND\text{-}GB}$ game. $\mathcal{C}$ returns GBPKE encryption of an arbitrary message $M_\gamma, \gamma \in_R \{0,1\}$ as $\langle C'_1, C'_2, C'_3, C'_4\rangle$. $\mathcal{B}$ rewrites the challenge ciphertext as $C^* = \langle (v^*)^{-1}C'_1, C'_2, (v^*)^{-1}C'_3, C'_4\rangle$ and sends it to $\mathcal{A}$. To be convinced that C^* is a valid encryption of M_γ under $\texttt{ID}^*$, recall that $d_{\texttt{ID}} = \langle d_{\texttt{ID1}}, d_{\texttt{ID2}}\rangle = \langle rsQ_{\texttt{ID}}, r^{-1}\rangle, H_1(\texttt{ID}^*) = Q^* = v^*Q_{\texttt{ID}}$ and corresponding secret key is $d^* = \langle d^*_1, d^*_2 = (r^*)^{-1}\rangle$. Now, $d^*_1 = r^*sQ^* = r^*s(v^*Q_{\texttt{ID}}) = r^{-1}r^*v^*(rsQ_{\texttt{ID}}) = r^{-1}r^*v^*d_{\texttt{ID1}}$

$$\begin{aligned}
[e((v^*)^{-1}C'_1, d^*_1)]^{d^*_2} &= [e((v^*)^{-1}C'_1, r^{-1}r^*v^*d_{\texttt{ID}})]^{(r^*)^{-1}} \\
&= e((v^*)^{-1}C'_1, r^{-1}v^*d_{\texttt{ID}}) \\
&= e((v^*)^{-1}C'_1, v^*d_{\texttt{ID}})^{r^{-1}} \\
&= e(C'_1, d_{\texttt{ID1}})^{d_{\texttt{ID2}}}
\end{aligned}$$

– **Phase 2 – Extraction Query**: Same as in **Phase 1**.

– **Phase 2 – Glassbox Decryption Query**: Same as in **Phase 1**. Only difference is, $\mathcal{B}$ aborts if the decryption of challenge ciphertext C^* is asked under the challenge identity $\texttt{ID}^*$. Attack on GBIBE fails prematurely.

– **Guess**: Eventually, $\mathcal{A}$ outputs its guess γ' for γ. $\mathcal{B}$ also emits the same bit as its output.

Unless aborted prematurely during simulation, algorithm $\mathcal{B}$ presents a proper $\mathsf{IND\text{-}ID\text{-}GB}$ game environment to $\mathcal{A}$. $\mathcal{B}$ controls the oracle $\mathcal{O}_{H1}$ itself. For the other three oracles, *viz.* $\mathcal{O}_{H2}, \mathcal{O}_{H3}$ and $\mathcal{O}_{H4}$, $\mathcal{B}$ cleverly answers the queries by appropriately rewriting certain parameters and forwarding those to respective oracle services provided by $\mathcal{C}$. Similarly, both secret key extraction and decrytion queries issued by $\mathcal{A}$ are answered by $\mathcal{B}$ itself, or by rerouting those to $\mathcal{C}$ depending on the identity $\texttt{ID}$ the query is being asked on. Moreover, if the game does not terminate until the challenge phase, $\mathcal{A}$ receives a well-formed GBIBE encrypted ciphertext of C^* of $M_\gamma, \gamma \in \{0,1\}$. Hence, by definition (as claimed in **Theorem 2**) of adversary $\mathcal{A}$, its advantage in $\mathsf{IND\text{-}ID\text{-}GB}$ game is $Adv_{\mathcal{A}} = |\Pr[\gamma = \gamma'] - \frac{1}{2}| \geq \nu(\kappa)$.

To attain a tighter reduction on the probability bound of $\mathcal{B}$'s advantage, we used partitioning of identity space via a biased coin toss with bias $\lambda = Pr[coin = 0], coin \in \{0,1\}$. Observe that,

– For $coin = 0$, $\mathcal{O}_{H1}$ returns a random $Q_i = v_iP_1 \in G_1$, for some $v_i \in_R \mathbb{Z}_p$. $\mathcal{B}$ can answer key extraction queries for such identities, but *aborts* if $\mathcal{A}$ throws challenge on this λ fraction of identities.

– For $coin = 0$, $\mathcal{O}_{H1}$ returns a random $Q_i = v_iQ_{\texttt{ID}} \in G_1$, for some $v_i \in_R \mathbb{Z}_p$. $\mathcal{B}$ can accept challenge for such identities, but *aborts* if $\mathcal{A}$ asks for key extraction on this $(1-\lambda)$ fraction of identities, in either phase 1 or phase 2.

Let us calculate the probability of having the game run till $\mathcal{A}$ emits a guess after asking at most q_X key extraction queries without getting $\mathcal{B}$ prematurely aborted. The probability of $\mathcal{B}$ not aborting during key extraction phase is λ^{q_X} and the same during challenge phase is $(1-\lambda)$. Composing both, the overall probability that $\mathcal{B}$ does not abort during simulation is $\mathrm{P}(\lambda) = \lambda^{q_X}(1-\lambda)$. Say, $\mathrm{P}(\lambda)$ attains maxima at λ_{max}.

Differentiating both sides,

$$\begin{aligned} \mathrm{P}(\lambda) &= \lambda^{q_X}(1-\lambda) \\ \mathrm{P}'(\lambda) &= q_X\lambda^{q_X-1}(1-\lambda) - \lambda^{q_X} \end{aligned}$$

Setting first derivative to zero,

$$\begin{aligned} q_X\lambda^{q_X-1}(1-\lambda_{max}) - \lambda_{max}^{q_X} &= 0 \\ \lambda_{max}^{q_X-1}\{q_X(1-\lambda_{max}) - \lambda_{max}\} &= 0 \\ \lambda_{max} &= \frac{q_X}{1+q_X} \end{aligned}$$

Therefore,

$$\begin{aligned} \mathrm{P}_{max} &= \lambda_{max}^{q_X}(1-\lambda_{max}) \\ &= \left(\frac{q_X}{1+q_X}\right)^{q_X}\left(1-\frac{q_X}{1+q_X}\right) \\ &= \left(\frac{1}{1+q_X}\right).\lim_{q_X\to\infty}\left(\frac{q_X}{1+q_X}\right)^{q_X} \\ &= \left(\frac{1}{1+q_X}\right).\frac{1}{\hat{e}} \qquad \text{... [}\hat{e}\text{ is the base of natural logarithm]} \end{aligned}$$

Hence, $\mathcal{B}$ solves the hard problem instance with an advantage at least $\dfrac{\nu(\kappa)}{\hat{e}\,(1+q_X)}$