Efficiently Obfuscating Re-Encryption Program under DDH Assumption

Akshayaram Srinivasan[1⋆] and C. Pandu Rangan[2]

[1] University of California, Berkeley, USA
Email: akshayaram@berkeley.edu
[2] Indian Institute of Technology Madras, India
Email: prangan55@gmail.com

Abstract. A re-encryption program (or a circuit) converts a ciphertext encrypted under Alice's public key pk_1 to a ciphertext of the same message encrypted under Bob's public key pk_2. Hohenberger et al. (TCC 2007) constructed a pairing-based obfuscator for a family of circuits implementing the re-encryption functionality under a new notion of obfuscation called as *average-case secure obfuscation*. Chandran et al. (PKC 2014) proposed a lattice-based construction for the same.
The construction given by Hohenberger et al. could only support encryptions of messages from a polynomial space and the decryption algorithm would have to perform a polynomial number of pairing operations in the worst case. The construction given by Chandran et al. satisfies only a relaxed notion of correctness.
In this work, we propose a simple and efficient obfuscator for the re-encryption functionality which not only supports encryption of messages from an exponential space but also involves only a constant number of group operations. Besides, our construction satisfies the strongest notion of correctness given by Hada (Eurocrypt 2010). We also strengthen the black-box security model for encryption - re-encryption system proposed by Hohenberger et al. and prove the average-case virtual black box property of our obfuscator as well as the security of our encryption - re-encryption system (in the strengthened model) under the DDH assumption. All our proofs are in the standard model.

Keywords: Re-encryption circuit, Average-case secure obfuscation, DDH Assumption, Standard Model

1 Introduction

A program obfuscator is a compiler which takes a program (or a circuit) as input and outputs an equivalent program which is "unintelligible". Barak, Goldreich, Impagliazzo, Sahai, Vadhan and Yang formalized the notion of program obfuscation in their seminal work [BGI+01]. According to their definition, an obfuscator must satisfy the following properties: preserves functionality, incurs a polynomial slowdown (when compared to the input program) and satisfies the *predicate black-box property*. The first property states that on all inputs, the output of the original program and the obfuscated program are exactly the same. The second property requires that the obfuscated program is not "too slow" when compared to the original program. Finally, the predicate black-box property stipulates that any predicate which is computable when given access to the obfuscated program must also be *efficiently* computable when given black-box access to the same program.

1.1 Average-case Secure Obfuscation

Hohenberger, Rothblum, shelat and Vaikuntanathan [HRsV07] noted that predicate black-box property [BGI+01] does not seem to give a meaningful security guarantee when the obfuscated functionality (like re-encryption) is a part of a larger cryptographic system (like the underlying encryption scheme protecting Alice and Bob's privacy). They demonstrated a need for a new definition of obfuscation which can be used to argue the security of cryptographic systems where the obfuscated functionality is only a sub-part.

⋆ Work done while the author was a student at Indian Institute of Technology Madras

To address the above issue, Hohenberger et al. proposed a new definition of obfuscation which they termed as *Average-case Secure Obfuscation*. This definition ensures that if there exists an adversary against a cryptographic scheme when given access to an obfuscated program then there exists an adversary with black box access to the original program with similar success probability provided the system has *distinguishable attack property* [HRsV07]. A cryptographic scheme is said to have distinguishable attack property if there exists a distinguisher which can "test" if a given algorithm can break the security of the scheme with just the public information and oracle access to the obfuscated program [3]. An informal template for proving the security of a scheme in this paradigm is as follows:

1. Prove that the scheme is secure against adversaries with black box access to a circuit $\mathcal{C}$ chosen uniformly at random from a circuit family C.
2. Prove that the scheme has distinguishable attack property.
3. Design an average-case secure obfuscator for the circuit family C.

1.2 Prior Work

Hohenberger et al. [HRsV07] designed an average case secure obfuscation for the re-encryption functionality under the Decision Linear and a strong variant of 3-party Decisional Diffie-Hellman assumption. The construction has a couple of drawbacks which severely limits its practical deployment. The first drawback is that it could only *support a message space of polynomial size.* This is because the running time of the decryption algorithm was directly proportional the size of the message space. Their decryption algorithm does an exhaustive search on the message space by computing a pairing operation on each message and then performing an equality check. Thus, it has to *compute a polynomial number of pairing operations in the worst case.* Moreover, the security of their construction was based on a strong assumption namely Strong 3-party DDH.

Remark 1. One can extend the system of [HRsV07] to message space of arbitrary size by using their construction for the message space $\{0, 1\}$. For an arbitrary message space $\mathcal{M}$, one can encrypt each message bit by bit and thus incurring a $O(\log(|\mathcal{M}|))$ overhead on encryption and decryption. For an exponential sized (in the security parameter λ) message space, the overhead on encryption and decryption algorithms would be $\mathsf{poly}(\lambda)$. But one would still like to have a system which performs constant number of operations (i.e have a constant overhead) in every algorithm.

Chandran, Chase, Liu, Nishimaki and Xagawa [CCL$^+$14] designed average case secure obfuscators for the re-encryption circuit under certain relaxed notions of correctness. In particular, they defined three relaxed notions of correctness. The first relaxation guarantees that the output of the original circuit and the obfuscated circuit are statistically close only on a subset of the actual inputs. The next relaxation guarantees that the output of the obfuscated program on a subset of inputs is correct with respect to some algorithm (like decryption). The final relaxation guarantees that the output of the obfuscated circuit and the original circuit are computationally indistinguishable.

A natural question which arises from the prior work is:

Does there exist an efficient obfuscator for re-encryption program under milder assumptions which satisfies the strongest notion of correctness, has a constant overhead in every algorithm and supports an exponential sized message space?

Proxy Re-Encryption A paradigm in cryptography which is closely related to re-encryption is *proxy re-encryption.* In a proxy re-encryption system, a semi-trusted proxy transforms ciphertexts intended for Alice (delegator) to a ciphertext of the same message for Bob (delegatee). Specifically, Alice provides the proxy

[3] Hohenberger et al. showed in [HRsV07] that cryptographic primitives like semantically secure encryption and re-encryption have this property.

with a re-key $RK_{A \to B}$ which is a function of her secret key sk_1 and Bob's public key pk_2. The proxy runs a specific algorithm (called as *re-encryption algorithm* in literature) which takes the ciphertext encrypted under Alice's public key and the re-key and outputs a ciphertext under Bob's public key. Proxy re-encryption schemes under strong notions of security have been proposed in [CH07],[LV08] and [CWYD10]. We note that none of the above results can be considered as obfuscations of re-encryption. In particular, giving the re-encryption key enables the proxy to learn some *non-black-box* information about the delegator's secret key. On the other hand, obfuscation of re-encryption circuit guarantees that *no* such non-black-box information is learned by the proxy.

1.3 Our Contributions

We highlight the main contributions of this work.

Main Result In this work, we propose a new encryption - re-encryption system that supports encryption of messages from an exponential (in security parameter) space, involves a constant number of group exponentiation operations in all algorithms and also design an average-case secure obfuscator for the re-encryption program which achieves the strongest notion of correctness (as in [Had10] and [HRsV07]). We prove the average case secure obfuscator property of our obfuscator and the security of our encryption system under the DDH assumption in the *standard model*. Informally, the main result in this work is:

Informal Theorem 1 *Under the DDH-assumption, there exists an average-case secure obfuscator for the family of circuits implementing the re-encryption functionality.*

Remark 2. We observe that our construction of obfuscator for re-encryption program is not secure when Bob has access to the obfuscated circuit. This is the case with all prior constructions of average-case secure obfuscators for re-encryption [HRsV07],[CCL+14] as well as encrypted signatures [Had10]. The construction is secure as long as the obfuscated circuit is run by some (possibly malicious) party other than Bob. It would be interesting to investigate the possibility of constructing average-case secure obfuscators which have "insider security". That is, they remain secure even when Bob has access to the obfuscated circuit. We leave this as an open problem.

Strengthening the Black-box Security model Recall that in order to design an obfuscator for re-encryption program in the paradigm of [HRsV07], one first needs to design an encryption - re-encryption system that is secure against adversaries given black box access to the re-encryption program. The security model considered by Hohenberger et al. [HRsV07] for this purpose is as follows: the challenger samples two public key-secret key pairs (pk_1, sk_1) and (pk_2, sk_2) and then provides pk_1, pk_2 to the adversary. The adversary (with oracle access to re-encryption program from pk_1 to pk_2) chooses two messages m_0 and m_1 and also gives information about the public key as well as the ciphertext level on which it wishes to be challenged. More precisely, the adversary can choose either to be challenged on a first level ciphertext [4] under pk_1 or a first level ciphertext under pk_2 or a second level ciphertext under pk_2 [5]. The scheme is secure if the adversary is unable to distinguish between the corresponding encryptions of m_0 and m_1.

We observe that the above security model is insufficient in properly capturing the full security notion of encryption - re-encryption system (See Remark 4). In particular, the above security model allows the following trivial but insecure encryption - re-encryption system to be secure. Consider any semantically secure encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$. To obtain a first level ciphertext of a message m under a public key pk, run the $\mathsf{Encrypt}$ algorithm on m and pk. The re-encryption program from pk_1 to pk_2 has sk_1, pk_1 and pk_2 hardwired into its description. When it is run with a first level ciphertext

[4] A first level ciphertext is the one which has not been re-encrypted. In other words, a first level ciphertext is given as input to the re-encryption program.

[5] A second level ciphertext under pk_2 is the output of re-encryption program from pk_1 to pk_2 on a first level ciphertext under pk_1

$c \leftarrow \mathsf{Encrypt}(m, pk_1)$, it decrypts the ciphertext using sk_1 and outputs $(\mathsf{Encrypt}(m, pk_1)||\mathsf{Encrypt}(sk_1, pk_2))$ where $||$ denotes concatenation. In order to decrypt a second level ciphertext, one can first decrypt the second component using sk_2 to obtain sk_1 using which one can decrypt the first component to obtain m. This system has an obvious drawback as it reveals sk_1 to the user with secret key sk_2. But one can prove that this system is secure under the security model considered in [HRsV07]. We also observe that it is possible to construct an average-case secure obfuscator for the above re-encryption program when one instantiates Π with a semantically secure encryption system which allows re-randomization of ciphertexts. The obfuscated program has $\mathsf{Encrypt}(sk_1, pk_2)$ hardcoded in its description and on any input $\mathsf{Encrypt}(m, pk_1)$, it re-randomizes the input ciphertext as well as the hardcoded value and outputs the concatenated encryptions. The average-case secure virtual black-box property follows from the semantic security of $\mathsf{Encrypt}$.

We strengthen the security model for encryption - re-encryption system as follows. We consider the security of the system under two different security games. The first game called as *Original Ciphertext Security* proceeds exactly as in [HRsV07] but the adversary is either challenged on a first level ciphertext under pk_1 or a first level ciphertext under pk_2. In the second game called as the *Transformed Ciphertext Security*, in addition to (pk_1, pk_2) the adversary is also provided with sk_1. In the challenge phase, the adversary obtains a second level ciphertext under pk_2 as the challenge ciphertext. Note that the above trivial encryption - re-encryption system is not transformed ciphertext secure as the adversary with access to sk_1 can directly decrypt the first component of the challenge message. We also observe that since the re-encryption program (which would later be obfuscated) makes use of sk_1 [6], it is natural to consider the security of output of re-encryption program against adversaries having access to sk_1.

Remark 3. Though this security model was not explicitly considered, all prior works [HRsV07], [CCV12], [CCL+14] satisfy this security notion.

Remark 4. We consider a stronger model for encryption - re-encryption security since the output of the re-encryption program (which we obfuscate in this work) does not have the same probability distribution as a fresh encryption of the message m under pk_2 (i.e as an output of another encryption algorithm $\mathsf{Encrypt2}(m, pk_2)$ as in [HRsV07]). If the output was distributed identically to a fresh encryption under pk_2 then the security model given by Hohenberger et al. is sufficient for our purposes [7]. The above discussion regarding the issues with the security model is for the generalized case where the output distribution of the re-encryption program and distribution of a freshly encrypted ciphertext under pk_2 are not identical. In fact, there are several (proxy) re-encryption systems (notably the one given in [CWYD10]) in literature which consider a generic case as above.

Overview of Construction The starting point of our construction is the semantically secure encryption scheme of El-Gamal [ElG85] instantiated over a group $\mathbb{G}$ of prime order p (where DDH is assumed to be true). Recall that in the El-Gamal scheme, the user keys are generated by choosing $x \xleftarrow{\$} \mathbb{Z}_p^*$ and setting the public key $pk = g^x$ and the secret key $sk = x$. In order to encrypt a message $m \in \mathbb{G}$, one chooses $r \xleftarrow{\$} \mathbb{Z}_p^*$ and outputs the ciphertext $[E_1, E_2] = (m \cdot (pk)^r, g^r)$. We consider a variant of the El-Gamal encryption scheme where the ciphertext is given by $[C_1, C_2] = (m \cdot g^r, (pk)^r)$. The decryption algorithm takes the secret key $sk = x$ and a ciphertext $[C_1, C_2] = (m \cdot g^r, (g^x)^r)$ and outputs $m = C_1 \cdot (C_2^{1/x})^{-1}$. It is easy to see that this variant is also semantically secure under the DDH-assumption.

Our goal is to obfuscate the trivial re-encryption program (which has sk_1 hardwired in its description) from pk_1 to pk_2. The main intuition behind our construction is illustrated through an initial attempt made by us which doesn't satisfy the definition of average case secure obfuscator but still captures the main idea in the final construction. The construction is inspired by the *proxy re-encryption* scheme given by Chow, Weng, Yang and Deng in [CWYD10].

[6] Note that it doesn't have access to sk_2

[7] The counter example discussed above will not work since the output of the re-encryption program is not identically distributed to a freshly generated ciphertext under pk_2

The re-encryption circuit $\mathcal{C}$ (which is to be obfuscated) has the secret key $sk_1 = x$ and the public key pk_2 hardwired in its description. It takes as input a ciphertext $c_1 = (m \cdot (g)^r, (g^x)^r)$ and first decrypts it using x to obtain m. It then chooses $h \xleftarrow{\$} \mathbb{G}$ and $r', v \xleftarrow{\$} \mathbb{Z}_p^*$ and outputs $(m \cdot g^{r'}, g^{H(h)r'}, h \cdot (g^y)^v, g^v)$ where H can (for now) be thought of as a Hash function from $\mathbb{G} \to \mathbb{Z}_p^*$. [8]

The obfuscator algorithm takes the above circuit $\mathcal{C}$ as input and reads the secret key sk_1 from the circuit description. It then outputs the description of the following circuit $\mathcal{C}'$. $\mathcal{C}'$ has three parameters $Z_1 = h \cdot (g^y)^v$, $Z_2 = g^v$ and $Z_3 = H(h)/x$ embedded in its description where $h \xleftarrow{\$} \mathbb{G}$, $v \xleftarrow{\$} \mathbb{Z}_p^*$. Note that Z_1, Z_2 is an El-Gamal encryption of h under pk_2. When given a ciphertext $[C_1, C_2] = (m \cdot g^r, (g^x)^r)$ as input, $\mathcal{C}'$ first re-randomizes the input ciphertext (to obtain $[C_1', C_2']$) as well as Z_1, Z_2 (to obtain $[Z_1', Z_2']$) by using freshly chosen random elements from $\mathbb{Z}_p^*$. It then computes $C_2'' = {C_2'}^{Z_3}$ and outputs $(C_1', C_2'', Z_1', Z_2')$. To decrypt, the holder of $sk_2 = y$, computes $h = Z_1' \cdot ({Z_2'}^y)^{-1}$ and then computes $m = C_1' \cdot ((C_2'')^{1/H(h)})^{-1}$.

The problem with the above construction is that the output of the obfuscated circuit $\mathcal{C}'$ does not have the same distribution as that of the original circuit $\mathcal{C}$. In particular, h is chosen uniformly at random for every invocation of $\mathcal{C}$ while it is not the case with $\mathcal{C}'$.

To fix the above problem, we define the randomness used in sampling the circuit $\mathcal{C}$ from the circuit family to not only include the random coins used to sample the keys for the system but also the random coins used in uniformly sampling a group element h. The circuit to be obfuscated will have sk_1, pk_2, and h hardwired into its description. When given a ciphertext $[C_1, C_2]$ as input, the program first decrypts the ciphertext using sk_1 to obtain m. It then chooses $r', v \xleftarrow{\$} \mathbb{Z}_p^*$ and outputs $[m \cdot g^{r'}, (g^{r'})^{H(h)}, h \cdot (g^y)^v, g^v]$ where h is the hardwired value [9]. The obfuscator algorithm is same as before except that the h used in computing Z_1 and Z_3 is read off from the circuit description. While this construction has the same output distribution as that of the original circuit, we still couldn't prove that the construction satisfies the average case secure virtual black box property. We slightly changed the output of the obfuscated circuit as well as the re-encryption program and arrived at our final construction.

The final construction is similar to the above attempt except that the obfuscated circuit (as well as the re-encryption program) chooses s uniformly at random from $\mathbb{Z}_p^*$ and outputs $[C_1', C_2'' \cdot (g^y)^s, Z_1', Z_2', g^s]$. We also instantiate H with a pseudo random generator. We could prove that this construction satisfies the average case virtual black box property under the DDH-assumption in the standard model.

2 Preliminaries

A function $\mu(\cdot) : \mathbb{N} \to \mathbb{R}^+$ is said to be **negligible**, if for every positive polynomial $p(\cdot)$, there exists an N such that for all $n \geq N$, $\mu(n) < 1/p(n)$. Given a probability distribution D on a universe U, we denote $x \leftarrow D$ as the operation of sampling an element x from U according to the distribution D. Given a finite set X, we use the notation $x \xleftarrow{\$} X$ for denoting the operation of sampling x from the set X uniformly. If two probability distributions D and D' defined on a set X are identical, we denote it by $D \approx D'$. If $\mathcal{A}$ is any probabilistic machine then $\mathcal{A}(x_1, \cdots, x_n)$ denotes the output distribution of $\mathcal{A}$. So, $x \leftarrow \mathcal{A}(x_1, x_2, \cdots, x_n)$ would denote the process of sampling x according to the output distribution of $\mathcal{A}$. Given n probability distributions $D_1, \cdots, D_n$, let $\{x_1 \leftarrow D_1; \cdots; x_n \leftarrow D_n : f(x_1, ..., x_n)\}$ be the probability distribution of a (possibly randomized) function f where the randomness in the distribution comes from the random coins used for sampling $x_1, \cdots, x_n$ according to the distributions $D_1, \cdots, D_n$ in the same order (as well as the random coins used by f if f is randomized). In a similar sense, we denote $Pr[x_1 \leftarrow D_1; \cdots; x_n \leftarrow D_n : E]$ as the probability of occurrence of an event E after the processes $x_1 \leftarrow D_1, \cdots, x_n \leftarrow D_n$ are performed in the same order and the probability is over the random coins used for sampling $x_1, \cdots, x_n$ according to the distributions $D_1, \cdots, D_n$ respectively. PPT machines refer to Probabilistic Polynomial Time Turing

[8] Note that the output of this re-encryption circuit is identically distributed to a freshly encrypted message under pk_2 using another encryption algorithm which works as described above.

[9] This would mean that the output of the re-encryption program will not be identically distributed as a fresh encryption under pk_2 and thus arises our need to consider a stronger security model.

machines. All PPT machines run in time polynomial in the security parameter denoted by λ. Sometimes, we also consider a non-uniform model of computation where the PPT machines may take an additional auxiliary input z of length polynomial in λ. If p is a prime number then let $\mathbb{Z}_p^*$ denote the set $\{1, 2, ..., p-1\}$.

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 $\{X_n\}_n \stackrel{c}{\approx} \{Y_n\}_n$, if for all non-uniform PPT machines (distinguishers) D, there exists a negligible function $\mu(\cdot)$ such that for all $n \in \mathbb{N}$, and for all $z \in \{0,1\}^{poly(n)}$,*

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

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

We now define the statistical distance between two random variables.

Definition 2 (Statistical Distance). *Let X and Y be two distributions on a finite set R. Then, the statistical distance between X and Y is defined as*

$$\Delta(X, Y) = \frac{1}{2}\sum_{r\in R}\left|Pr[c \leftarrow X : c = r] - Pr[c \leftarrow Y : c = r]\right|$$

We state and prove this following simple Lemma which will be used through out the work.

Lemma 1. *For all distributions X_n and Y_n, for all PPT distinguishers D and for all $z \in \{0,1\}^{poly(n)}$ we have,*

$$\Delta(D(X_n, z), D(Y_n, z)) = \left|Pr[b \leftarrow D(X_n, z) : b = 1] - Pr[b \leftarrow D(Y_n, z) : b = 1]\right|$$

Proof. The lemma directly follows from the following observation that, $\left|Pr[b \leftarrow D(X_n, z) : b = 1] - Pr[b \leftarrow D(Y_n, z) : b = 1]\right|$ and $\left|Pr[b \leftarrow D(X_n, z) : b = 0] - Pr[b \leftarrow D(Y_n, z) : b = 0]\right|$ are equal since $D(X_n, z)$ and $D(Y_n, z)$ are distributions on $\{0, 1\}$. The statement of the lemma can then be derived using the definition for statistical distance. □

This lemma implies that $\{X_n\}_n \stackrel{c}{\approx} \{Y_n\}_n$ if and only if $D(X_n, z)$ and $D(Y_n, z)$ are statistically close for all PPT distinguishers D and for all auxiliary input z.

We now recall the Decisional Diffie-Hellman (DDH) assumption on prime order groups. Let Gen be an algorithm which takes 1^λ as input and randomly generates the parameters $(p, \mathbb{G}, g)$ where p is a λ bit prime, $\mathbb{G}$ is a multiplicative group of order p and g is a generator for $\mathbb{G}$.

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

$$\left\{\begin{array}{l}(p, \mathbb{G}, g) \leftarrow \mathsf{Gen}(1^\lambda); \\ a, b \stackrel{\$}{\leftarrow} \mathbb{Z}_p^* : \\ (g, g^a, g^b, g^{ab})\end{array}\right\}_\lambda \stackrel{c}{\approx} \left\{\begin{array}{l}(p, \mathbb{G}, g) \leftarrow \mathsf{Gen}(1^\lambda); \\ a, b, c \stackrel{\$}{\leftarrow} \mathbb{Z}_p^* : \\ (g, g^a, g^b, g^c)\end{array}\right\}_\lambda$$

We recall the syntax and security notions (single message and multi message security) for a Public Key Encryption (PKE) system in Appendix A.

We formally describe the El-Gamal encryption system and its variant in Appendix B.

We state the theorems regarding the single message security and the multi-message security of El-Gamal encryption system and its variant.

Theorem 1 (Single message security). *Assuming that DDH-assumption holds in the group $\mathbb{G}$, both El-Gamal encryption and its variant are single message secure.*

Theorem 2 (Multi message security). *Assuming the DDH-assumption holds in the group $\mathbb{G}$, both El-Gamal encryption and its variant are multi-message secure.*

We now recall the definition of Pseudo Random Generator (PRG) from [Gol01].

Definition 4. *A Pseudo Random Generator G is a deterministic polynomial time algorithm satisfying the following two conditions:*

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 Average-case Secure Obfuscation

Let $C = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of polynomial sized circuits. For a length parameter λ, let $\mathcal{C}_\lambda$ be the set of circuits in C with input length $p_{in}(\lambda)$ and output length $p_{out}(\lambda)$ where $p_{in}(\cdot)$ and $p_{out}(\cdot)$ are polynomials. The circuit family C has an associated sampling algorithm Samp which takes 1^λ as input and outputs a circuit $\mathcal{C}$ chosen uniformly at random from $\mathcal{C}_\lambda$. We also assume that there exists efficient ($\mathsf{Encode}, \mathsf{Decode}$) algorithms which encodes and decodes a given circuit $\mathcal{C}$ into binary strings when used as input/output of Turing machines. We make implicit use of such encoding and decoding algorithms and do not mention them explicitly.

We use similar notations (with some minor changes) as in [Had10] to denote probabilistic circuits. A probabilistic circuit $\mathcal{C}(x;r)$ takes two inputs. The first input is called as the *regular input* and the second input is termed as the *random input*. The output of a probabilistic circuit on a regular input (denoted by $\mathcal{C}(x;\cdot)$) can be viewed as a probability distribution where the randomness in the distribution comes from the random choice of r. We say that a machine $\mathcal{A}$ has *oracle access* to a probabilistic circuit $\mathcal{C}$ (denoted by $\mathcal{A}^{\mathcal{O}(\mathcal{C})}$) if during the oracle queries, $\mathcal{A}$ can only specify the regular input x to the circuit and the random input r is chosen uniformly at random from the corresponding sample space by the oracle $\mathcal{O}$. In other words, $\mathcal{A}$ gets a random sample from the distribution $\mathcal{C}(x;\cdot)$ for every oracle query x. The output of a probabilistic machine $\mathcal{A}$ having oracle access to a probabilistic circuit $\mathcal{C}$ (denoted by $\mathcal{A}^{\mathcal{O}(\mathcal{C})}(x_1,\cdots,x_n)$) is a probability distribution where the randomness in the distribution comes from the random coins used by $\mathcal{A}$ as well as the random coins used by $\mathcal{O}$ in answering $\mathcal{A}$'s oracle queries. We say that $\mathcal{B}$ *evaluates* a probabilistic circuit $\mathcal{C}$ (or in other words, $\mathcal{B}$ is an evaluator of $\mathcal{C}$) on regular input x, if $\mathcal{B}$ supplies the regular input as well as the random input r chosen uniformly at random from the corresponding sample space and outputs $\mathcal{C}(x;r)$. We assume that there exists efficient algorithms that take the description of a PPT machine M as input and outputs the description of a probabilistic circuit $\mathcal{C}$ having the same functionality as M. We use $|\mathcal{C}|$ to denote the size of a circuit $\mathcal{C}$.

We recall the notion of *average case secure obfuscation* given in [HRsV07].

Definition 5 ([HRsV07],[Had10]). *A PPT machine Obf that takes as input a (probabilistic) circuit and outputs a new (probabilistic) circuit is an average-case secure obfuscator for the circuit family $C = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ with an associated sampling algorithm Samp if it satisfies the following properties:*

1. *Preserving Functionality: For all length parameter $\lambda \in \mathbb{N}$ and for all $\mathcal{C} \in \mathcal{C}_\lambda$:*

$$Pr[\mathcal{C}' \leftarrow \mathsf{Obf}(\mathcal{C}) : \exists x \in \{0,1\}^{p_{in}(\lambda)}, \Delta\big(\mathcal{C}'(x;\cdot), \mathcal{C}(x;\cdot)\big) \neq 0] = 0$$

2. *Polynomial Slowdown: There exists a polynomial $p(\cdot)$ such that for sufficiently large length parameters λ, for any $\mathcal{C} \in \mathcal{C}_\lambda$, we have*

$$Pr[\mathcal{C}' \leftarrow \mathsf{Obf}(\mathcal{C}) : |\mathcal{C}'| \leq p(|\mathcal{C}|)] = 1$$

3. *Average-case Secure Virtual Black Box: There exists a PPT machine (simulator)* Sim *such that for every PPT distinguisher D, there exists a negligible function* $\mathsf{neg}(\cdot)$ *such that for every length parameter λ and for every $z \in \{0,1\}^{poly(\lambda)}$:*

$$\left| Pr\begin{bmatrix} \mathcal{C} \leftarrow \mathsf{Samp}(1^\lambda); \\ \mathcal{C}' \leftarrow \mathsf{Obf}(\mathcal{C}); \\ b \leftarrow D^{\mathcal{O}(\mathcal{C})}(\mathcal{C}', z): \\ b = 1 \end{bmatrix} - Pr\begin{bmatrix} \mathcal{C} \leftarrow \mathsf{Samp}(1^\lambda); \\ \mathcal{C}' \leftarrow \mathsf{Sim}^{\mathcal{O}(\mathcal{C})}(1^\lambda, z); \\ b \leftarrow D^{\mathcal{O}(\mathcal{C})}(\mathcal{C}', z): \\ b = 1 \end{bmatrix} \right| \leq \mathsf{neg}(\lambda)$$

Remark 5. [Had10] The definition given in [HRsV07] considers a relaxed notion of correctness. Specifically, it allows a the output distribution of the obfuscated circuit and the original circuit to have a negligible statistical distance with a negligible probability. Here, we consider a stronger notion of correctness where we require that the output distribution of the original circuit and the obfuscated circuit to be identical.

3 Obfuscator for Re-encryption Functionality

In this section, we describe our new encryption system, the re-encryption functionality which is to be obfuscated and finally the construction of an average case secure obfuscator for the functionality.

3.1 New Encryption Scheme

The new encryption system under consideration is same as the El-Gamal system variant described in Appendix 2 with some minor modifications in the Setup algorithm.

New Encryption Scheme

- $\mathsf{Setup}(1^\lambda)$: Let $(p, \mathbb{G}, g) \leftarrow \mathsf{Gen}(1^\lambda)$. Let H be a pseudo random generator which takes as input an element from $\mathbb{G}$ and outputs an element in $\mathbb{Z}_p^*$.[a] Output the public parameters as $params = (p, g, \mathbb{G}, H)$ with message space $\mathcal{M} = \mathbb{G}$.
- $\mathsf{KeyGen}(1^\lambda, params)$: Choose $x \xleftarrow{\$} \mathbb{Z}_p^*$ and set the public key pk to be (g, g^x) and the secret key $sk = x$.
- $\mathsf{Encrypt1}(m, pk)$: Parse pk as (g, g^x). Choose a random $r \xleftarrow{\$} \mathbb{Z}_p^*$ and output $(m \cdot g^r, (g^x)^r)$.
- $\mathsf{Decrypt1}(sk, [C_1, C_2])$: Parse the secret key sk as x. Output $m = (C_1) \cdot ((C_2)^{1/x})^{-1}$.

[a] The definition of pseudo random generator given in this work assumes the domain and the range to be bit strings. We note that it can be extended to any domain and range assuming efficient encoding and decoding functions from the domain to bit strings and from bit strings to range. The expansion factor of H depends on the actual encoding and decoding schemes used.

3.2 Re-encryption functionality

Let (pk_1, sk_1) and (pk_2, sk_2) be two key-pairs which are obtained by running the KeyGen algorithm with independent random tapes. Let $h \xleftarrow{\$} \mathbb{G}$ be an element chosen uniformly and independently at random from the group $\mathbb{G}$. The PPT algorithm performing re-encryption from pk_1 to pk_2 (denoted by $\mathsf{Re-Enc}_{1\to 2}$) is described below.

$\mathsf{Re-Enc}_{1\to 2}$

Input: $c_1 = [C_1, C_2]$ or special symbol denoted by keys[a]
Constants:[b] $sk_1 = x$, $pk_1 = (g, g^x)$, $pk_2 = (g, g^y)$ and h

1. If $input = \mathsf{keys}$, output (pk_1, pk_2).

2. Else,
   - Compute $m = \mathsf{Decrypt1}(sk_1, c_1)$.
   - Choose $r', v, s \xleftarrow{\$} \mathbb{Z}_p^*$.
   - Output $[C_1', C_2', C_3', C_4', C_5'] = [m \cdot g^{r'}, (g^{H(h)})^{r'} \cdot (g^y)^s, h \cdot (g^y)^v, g^v, g^s]$.

[a] $\mathsf{keys} \notin \mathbb{G} \times \mathbb{G}$. We need this for a technical part in the proof.
[b] Constants in a program denotes those values which are hardcoded in the program description

Re-encryption Circuit Family Let $\mathcal{C}_{sk_1,pk_1,pk_2,h}$ be the description of a probabilistic circuit implementing the program $\mathsf{Re}-\mathsf{Enc}_{1\to 2}$. We note that the constants in the above program are hardwired in the circuit description. These constants can be extracted when given access to the description of the circuit. Formally, the class of circuits implementing the re-encryption functionality for a given length parameter λ is,

$$\mathcal{C}_\lambda = \{\mathcal{C}_{sk_1,pk_1,pk_2,h} : (pk_1, sk_1) \leftarrow \mathsf{KeyGen}(1^\lambda), (pk_2, sk_2) \leftarrow \mathsf{KeyGen}(1^\lambda), h \xleftarrow{\$} \mathbb{G}\}$$

The circuit family implementing the re-encryption functionality is given by $C = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$. The associated sampling algorithm Samp which samples a circuit $\mathcal{C}$ uniformly at random from $\mathcal{C}_\lambda$ is described below:

Samp

Input: 1^λ

1. $(p, \mathbb{G}, g, H) \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $h \xleftarrow{\$} \mathbb{G}$.
3. Output the circuit description of $C_{sk_1,pk_1,pk_2,h}$.

The evaluator of the circuit $\mathcal{C}_{sk_1,pk_1,pk_2,h}$ supplies the regular input which is either the ciphertext $c_1 = [C_1, C_2]$ or the special symbol keys and also supplies the random input rand chosen uniformly at random from $\{0,1\}^{3\lambda}$ to the circuit for sampling r', v, s uniformly from $\mathbb{Z}_p^*$.

Decrypting the Circuit Output The output of $\mathsf{Re}-\mathsf{Enc}_{1\to 2}$ can be decrypted using the following algorithm $\mathsf{Decrypt2}$:

Decrypt2

Input: $sk_2, [C_1', C_2', C_3', C_4', C_5']$:

1. Parse sk_2 as y.
2. Compute $h = C_3' \cdot ({C_4'}^y)^{-1}$.
3. Compute $C_2'' = C_2' \cdot ((C_5')^y)^{-1}$
4. Output $m = (C_1') \cdot ((C_2'')^{1/(H(h))})^{-1}$.

3.3 Obfuscator construction

We now present the construction of an average-case secure obfuscator (denoted by Obf) for the re-encryption circuit family defined in Section 3.2.

Obf

Input: $\mathcal{C}_{sk_1,pk_1,pk_2,h}$

1. Read $sk_1 = x$, $pk_1 = (g, g^x)$, $pk_2 = (g, g^y)$ and h from the description of the circuit $\mathcal{C}_{sk_1,pk_1,pk_2,h}$.

2. Select $v \xleftarrow{\$} \mathbb{Z}_p^*$.
3. Compute $(Z_1, Z_2, Z_3) = (h \cdot (g^y)^v, g^v, H(h)/x)$.
4. Output the description of a circuit implementing the program $\mathsf{Re} - \mathsf{Enc}'_{1\to 2}$ described below with $pk_1, pk_2, Z_1, Z_2, Z_3$ as the constants in the program.

$\mathsf{Re} - \mathsf{Enc}'_{1\to 2}$

Input: $c_1 = [C_1, C_2]$ or special symbol denoted by keys.
Constants: $pk_1, pk_2, Z_1, Z_2, Z_3$.

1. If $input = \mathsf{keys}$, output (pk_1, pk_2).
2. Else,
   - Choose two re-randomization values $r', v' \xleftarrow{\$} \mathbb{Z}_p^*$.
   - Re-randomize the input as $C'_1 = C_1 \cdot g^{r'}$, $\overline{C_2} = (C_2 \cdot (g^x)^{r'})$ and the hardwired values as $C'_3 = Z_1 \cdot (g^y)^{v'}$, $C'_4 = Z_2 \cdot g^{v'}$.
   - Compute $\overline{\overline{C_2}} = (\overline{C_2})^{Z_3}$.
   - Choose $s \xleftarrow{\$} \mathbb{Z}_p^*$.
   - Compute $C'_2 = \overline{\overline{C_2}} \cdot (g^y)^s$ and $C'_5 = g^s$
   - Output $[C'_1, C'_2, C'_3, C'_4, C'_5]$.

Let $\mathcal{C}'$ denote the circuit implementing $\mathsf{Re} - \mathsf{Enc}'_{1\to 2}$. The evaluator for the circuit $\mathcal{C}'$ provides either $c_1 = [C_1, C_2]$ or special symbol keys as the regular input and $\mathsf{rand} \xleftarrow{\$} \{0,1\}^{3\lambda}$ as the random input for sampling r', v', s uniformly from $\mathbb{Z}_p^*$.

Remark 6. The obfuscated circuit $\mathcal{C}'$ is generated by the owner of sk_1 but can be evaluated by anyone. We assume (as described in Remark 2) that the evaluator of $\mathcal{C}'$ and the owner of sk_2 do not collude.

4 Security of New Encryption Scheme

We now describe the security model for semantic security of the encryption scheme when the adversary is given black box access to re-encryption functionality. In view of discussion presented in Section 1.3, we modify the security model given in [HRsV07] as follows.

4.1 Security Model

Let $\mathcal{C} \leftarrow \mathsf{Samp}(1^\lambda)$ [10] be the re-encryption circuit from pk_1 to pk_2.

- **Original Ciphertext Security:** Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against the original ciphertext security.

  Definition 6. *Let $\mathit{\Pi}$ be an encryption scheme and let $IND_{b,ori}(\mathit{\Pi}, \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2), \lambda, i)$ where $b \in \{0,1\}$ and $i \in \{1,2\}$, denote the following experiment:*

  $IND_{b,ori}(\mathit{\Pi}, \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2), \lambda, i)$

  1. $params \leftarrow \mathsf{Setup}(1^\lambda)$
  2. $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(params)$ *and* $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}(params)$.
  3. *Choose* $h \xleftarrow{\$} \mathbb{G}$.
  4. *Set* $\mathcal{C} = \mathcal{C}_{sk_1, pk_1, pk_2, h}$ [a].

[10] For the ease of exposition, we drop the subscripts sk_1, pk_1, pk_2, h.

5. $(m_0, m_1, state) \leftarrow \mathcal{A}_1^{\mathcal{O}(\mathcal{C})}(pk_1, pk_2, params)$.
6. $C^* \leftarrow \mathsf{Encrypt1}(m_b, pk_i, params)$.
7. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}(\mathcal{C})}(C^*, state)$.
8. *Output* b'

[a] Note that setting $\mathcal{C}$ in this way is equivalent to sampling $\mathcal{C}$ using the Samp algorithm

The scheme Π is said to be ***original ciphertext secure*** *with respect to the oracle access to $\mathcal{C}$ if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and for all $i \in \{1, 2\}$, there exists a negligible function $\mu(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\Delta\big(IND_{0,ori}(\Pi, \mathcal{A}, \lambda, i), IND_{1,ori}(\Pi, \mathcal{A}, \lambda, i)\big) \leq \mu(\lambda)$$

– **Transformed Ciphertext Security**: Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against the transformed ciphertext security.

Definition 7. *Let Π be an encryption scheme and let $IND_{b,tran}(\Pi, \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2), \lambda)$ where $b \in \{0, 1\}$ denote the following experiment:*

$IND_{b,tran}(\Pi, \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2), \lambda)$

1. $params \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(params)$ *and* $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}(params)$.
3. *Choose* $h \xleftarrow{\$} \mathbb{G}$.
4. *Set* $\mathcal{C} = \mathcal{C}_{sk_1, pk_1, pk_2, h}$.
5. $(m_0, m_1, state) \leftarrow \mathcal{A}_1^{\mathcal{O}(\mathcal{C})}(params, pk_1, pk_2, sk_1)$.
6. $\mathsf{rand} \xleftarrow{\$} \{0, 1\}^{3\lambda}$
7. $C^* \leftarrow \mathcal{C}(\mathsf{Encrypt1}(m_b, pk_1, params); \mathsf{rand})$.
8. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}(\mathcal{C})}(C^*, state)$.
9. *Output* b'

The scheme Π is said to be ***transformed ciphertext secure*** *with respect to the oracle access to $\mathcal{C}$ if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mu(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\Delta\big(IND_{0,tran}(\Pi, \mathcal{A}, \lambda), IND_{1,tran}(\Pi, \mathcal{A}, \lambda)\big) \leq \mu(\lambda)$$

4.2 Security Proof

We now show that the New Encryption Scheme is both *original ciphertext secure* as well as *transformed ciphertext secure.*

Theorem 3. *The New Encryption Scheme is* ***original ciphertext secure*** *with respect to the oracle $\mathcal{C}_{sk_1, pk_1, pk_2, h}$ under the DDH-assumption.*

Proof. We will consider the cases $i = 1$ and $i = 2$ separately.
Case-I: $i = 1$.

Assume for the sake of contradiction that there exists an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the original ciphertext security of the New Encryption System such that,

$$\Delta\big(IND_{0,ori}(\Pi, \mathcal{A}, \lambda, 1), IND_{1,ori}(\Pi, \mathcal{A}, \lambda, 1)\big) = \delta$$

where δ is non-negligible. We now construct a polynomial time algorithm $\mathcal{B}$ that solves the DDH problem with non-negligible advantage.

$\mathcal{B}$ receives the tuple (g, g^x, g^r, Q) from the DDH-challenger. It chooses a random $y \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $pk_1 = (g, g^x)$ and $pk_2 = (g, g^y)$. $\mathcal{B}$ chooses $h \xleftarrow{\$} \mathbb{G}$ and $v \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $Z_1 = h \cdot (g^y)^v$, $Z_2 = g^v$ and $Z_3' \xleftarrow{\$} \mathbb{Z}_p^*$. $\mathcal{B}$ provides $(pk_1, pk_2, params)$ to $\mathcal{A}_1$.

$\mathcal{B}$ simulates the oracle access of $\mathcal{C}_{sk_1, pk_1, pk_2, h}$ to $\mathcal{A}_1$ and $\mathcal{A}_2$ as follows: $\mathcal{B}$ runs the program $\mathsf{Re-Enc}'_{1\to 2}$ described in Section 3.3 with the values pk_1, pk_2, Z_1, Z_2,
Z_3' as constants. It remains to show that $\mathcal{A}_1$ and $\mathcal{A}_2$ would not be able to distinguish $\mathcal{B}$'s simulation and that of the original circuit. This follows directly from the following Lemma.

Lemma 2.

$$\left\{\begin{array}{l} (p, g, \mathbb{G}, H) \leftarrow \mathsf{Setup}(1^\lambda); \\ x, y \xleftarrow{\$} \mathbb{Z}_p^*; \\ pk_1 = (g, g^x); \\ pk_2 = (g, g^y); \\ h \xleftarrow{\$} \mathbb{G}; \\ Z_3', v \xleftarrow{\$} \mathbb{Z}_p^* : \\ \big(pk_1, pk_2, h \cdot (g^y)^v, g^v, Z_3'\big) \end{array}\right\}_\lambda \overset{c}{\approx} \left\{\begin{array}{l} (p, g, \mathbb{G}, H) \leftarrow \mathsf{Setup}(1^\lambda); \\ x, y \xleftarrow{\$} \mathbb{Z}_p^*; \\ pk_1 = (g, g^x); \\ pk_2 = (g, g^y); \\ h \xleftarrow{\$} \mathbb{G}; \\ v \xleftarrow{\$} \mathbb{Z}_p^* : \\ \big(pk_1, pk_2, h \cdot (g^y)^v, g^v, H(h)/x\big) \end{array}\right\}_\lambda$$

Proof. We show that the $\mathsf{Hyb0}$ is computationally indistinguishable to $\mathsf{Hyb1}$ which is in turn indistinguishable to $\mathsf{Hyb2}$.

$$\mathsf{Hyb0} \approx \left\{\begin{array}{l} (p, g, \mathbb{G}, H) \leftarrow \mathsf{Setup}(1^\lambda); \\ x, y \xleftarrow{\$} \mathbb{Z}_p^*; \\ pk_1 = (g, g^x); \\ pk_2 = (g, g^y); \\ h \xleftarrow{\$} \mathbb{G}; \\ Z_3', v \xleftarrow{\$} \mathbb{Z}_p^* : \\ \big(pk_1, pk_2, h \cdot (g^y)^v, g^v, Z_3'\big) \end{array}\right\}_\lambda$$

$$\mathsf{Hyb1} \approx \left\{\begin{array}{l} (p, g, \mathbb{G}, H) \leftarrow \mathsf{Setup}(1^\lambda); \\ x, y \xleftarrow{\$} \mathbb{Z}_p^*; \\ pk_1 = (g, g^x); \\ pk_2 = (g, g^y); \\ h, h' \xleftarrow{\$} \mathbb{G}; \\ v \xleftarrow{\$} \mathbb{Z}_p^* : \\ \big(pk_1, pk_2, h \cdot (g^y)^v, g^v, H(h')/x\big) \end{array}\right\}_\lambda$$

$$\mathsf{Hyb2} \approx \left\{\begin{array}{l} (p, g, \mathbb{G}, H) \leftarrow \mathsf{Setup}(1^\lambda); \\ x, y \xleftarrow{\$} \mathbb{Z}_p^*; \\ pk_1 = (g, g^x); \\ pk_2 = (g, g^y); \\ h \xleftarrow{\$} \mathbb{G}; \\ v \xleftarrow{\$} \mathbb{Z}_p^* : \\ \big(pk_1, pk_2, h \cdot (g^y)^v, g^v, H(h)/x\big) \end{array}\right\}_\lambda$$

Claim. Assuming that H is a pseudo random generator, $\mathsf{Hyb0}$ and $\mathsf{Hyb1}$ are computationally indistinguishable.

Proof. Suppose there exists an polynomial time adversary $\mathcal{A}$ distinguishing $\mathsf{Hyb0}$ from $\mathsf{Hyb1}$ with a non-negligible advantage, we construct a polynomial time adversary $\mathcal{B}$ distinguishing the output of the pseudo random generator H from the uniform distribution. $\mathcal{B}$ chooses $x, y \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $pk_1 = (g, g^x)$, $pk_2 = (g, g^y)$. It chooses $h \xleftarrow{\$} \mathbb{G}$ and $v \xleftarrow{\$} \mathbb{Z}_p^*$. It computes $h \cdot (g^y)^v$ and g^v. It receives the challenge Q and auxiliary information z from the challenger for the pseudo random generator game. $\mathcal{B}$ runs $\mathcal{A}$ with $\big(pk_1, pk_2, h \cdot (g^y)^v, g^v, Q/x, z\big)$ as input. We can easily see that if Q was the output of the pseudo random generator on a random group element then the input distribution to $\mathcal{A}$ is identically distributed as that of $\mathsf{Hyb1}$. Else, it is identically distributed as in $\mathsf{Hyb0}$. $\mathcal{B}$ outputs the same bit as $\mathcal{A}$ does. Hence, the advantage of $\mathcal{B}$ in the game against the pseudo random generator challenger is same as the advantage $\mathcal{A}$ has in distinguishing between the two hybrids. Hence, $\mathsf{Hyb0}$ and $\mathsf{Hyb1}$ are computationally indistinguishable. ◻

Claim. Assuming the single message security (Theorem 2) of El-Gamal encryption scheme, $\mathsf{Hyb1}$ and $\mathsf{Hyb2}$ are computationally indistinguishable.

Proof. Suppose there exists an algorithm $\mathcal{A}$ that can distinguish with non-negligible advantage between $\mathsf{Hyb1}$ and $\mathsf{Hyb2}$, we construct an adversary $\mathcal{B}$ against the single message security of the El-Gamal encryption with the same non-negligible advantage.

$\mathcal{B}$ receives the public key g^y from the El-Gamal challenger. It chooses $x \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $pk_1 = (g, g^x)$ and $pk_2 = (g, g^y)$. It chooses random $h, h' \xleftarrow{\$} \mathbb{G}$ (independently) and gives h, h' as the challenge messages to the El-Gamal challenger. It receives a challenge ciphertext $C^* = (C_1^*, C_2^*)$ and auxiliary information z. It then runs $\mathcal{A}$ with $\big(pk_1, pk_2, C_1^*, C_2^*, H(h)/x, z\big)$ as input. It is easy to the see that when the challenge ciphertext is an encryption of h, the input to $\mathcal{A}$ is identically distributed to $\mathsf{Hyb2}$ and when it is an encryption of h', the input to $\mathcal{A}$ is identically distributed to $\mathsf{Hyb1}$.

$\mathcal{B}$ outputs the same bit as $\mathcal{A}$ outputs. It is easy to see that the advantage that $\mathcal{B}$ has in distinguishing between the ciphertexts of the challenge messages is same as the advantage that $\mathcal{A}$ has in distinguishing between the two hybrids. Hence, by assumption $\mathcal{B}$ has a non-negligible advantage in the El-Gamal security game which is a contradiction. ◻

From the above two claims, we can infer that $\mathsf{Hyb0}$ and $\mathsf{Hyb2}$ are computationally indistinguishable. ◻

The above lemma shows that any polynomial time adversary would not be able to distinguish between the cases when Z_3' is chosen uniformly at random and the case when it is set to $H(h)/x$. We observe that the $\mathsf{Re}-\mathsf{Enc}'_{1\to 2}(pk_1, pk_2, Z_1,$
$Z_2, H(h)/x)$ preserves the same functionality as that of the original circuit (proved in Appendix 4). Note that the functionality of $\mathsf{Re}-\mathsf{Enc}'_{1\to 2}$ is dependent only on the constants which are hardcoded. That is, given the constants, $\mathsf{Re}-\mathsf{Enc}'_{1\to 2}$ can be run by a PPT machine on various inputs producing identical output distribution as that of the original program. The lemma states that no adversary would be able to distinguish between the constants which are hardcoded in the original and the simulated program. This implies that no adversary would be able to distinguish the outputs of the original and the simulated programs on specific inputs. Hence, as a consequence of the above lemma $\mathcal{A}_1$ or $\mathcal{A}_2$ would not be able to distinguish between oracle access to $\mathsf{Re}-\mathsf{Enc}'_{1\to 2}(pk_1, pk_2, Z_1, Z_2, H(h)/x)$ and $\mathsf{Re}-\mathsf{Enc}'_{1\to 2}(pk_1, pk_2, Z_1, Z_2, Z_3')$ except with negligible probability. Let that negligible probability be denoted by δ'.

$\mathcal{A}_1$ outputs two messages m_0 and m_1. $\mathcal{B}$ tosses a random coin $\beta \in \{0, 1\}$ and gives the challenge ciphertext $C^* = (m_\beta \cdot g^r, Q)$. $\mathcal{A}_2$ finally outputs its guess β' of β. If $\beta = \beta'$ then $\mathcal{B}$ outputs 1 meaning that Q is a DDH-instance. Else, $\mathcal{B}$ outputs 0.

We first observe that if Q was a DDH-instance then, $\mathcal{B}$ outputs the challenge ciphertext which is identically distributed to the original ciphertext. Also it gives a simulation of the re-encryption oracle which is computationally indistinguishable from the original oracle. Hence, the probability that $\beta = \beta'$ in this case is given by $1/2 + \delta - \delta'$.

If Q was not a DDH instance, then due to random choice of Q and the random choice of r the probability that $\beta' = \beta$ is $1/2$.

Hence, the advantage of $\mathcal{B}$ against the DDH game is given by $\delta - \delta'$. We have assumed δ to be non-negligible and by the above lemma δ' is negligible. Therefore, $\mathcal{B}$ has a non-negligible advantage against the DDH challenger which is a contradiction. This completes the proof for case $i = 1$.

Case-II: $i = 2$.

Let us assume for the sake of contradiction that there exists PPT algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that:

$$\Delta\big(IND_{0,ori}(\Pi, \mathcal{A}, \lambda, 2), IND_{1,ori}(\Pi, \mathcal{A}, \lambda, 2)\big) = \delta$$

where δ is non-negligible. We now construct a polynomial time algorithm $\mathcal{B}$ that solves the DDH problem with non-negligible advantage.

$\mathcal{B}$ receives the tuple (g, g^y, g^r, Q) from the DDH-challenger. It chooses a random $x \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $pk_1 = (g, g^x)$ and $pk_2 = (g, g^y)$. $\mathcal{B}$ chooses $h \xleftarrow{\$} \mathbb{G}$.

$\mathcal{B}$ simulates the oracle access of $\mathcal{C}_{sk_1,pk_1,pk_2,h}$ to $\mathcal{A}_1$ and $\mathcal{A}_2$ as follows: $\mathcal{B}$ runs the program $\mathsf{Re}-\mathsf{Enc}_{1\to 2}$ described in Section 3.3 with the values x, pk_1, pk_2, h as constants. Thus, the oracle access to $\mathcal{C}_{sk_1,pk_1,pk_2,h}$ is simulated perfectly by $\mathcal{B}$.

$\mathcal{A}_1$ outputs two messages m_0 and m_1. $\mathcal{B}$ tosses a random coin $\beta \in \{0, 1\}$ and gives the challenge ciphertext $C^* = (m_\beta \cdot g^r, Q)$. $\mathcal{A}_2$ finally outputs its guess β' of β. If $\beta = \beta'$ then $\mathcal{B}$ outputs 1 meaning that Q is a DDH-instance. Else, $\mathcal{B}$ outputs 0.

It is easy to see that if Q is a DDH-instance then the probability that $\beta = \beta'$ is given by $\delta + 1/2$. Else, due to randomness of Q and the random choice of r, the probability that $\beta = \beta'$ is exactly 1/2. Therefore, the advantage of $\mathcal{B}$ against the DDH-challenger is δ which is non-negligible. Hence, we have arrived at a contradiction.

□

Theorem 4. *The New Encryption Scheme is* ***transformed ciphertext secure*** *with respect to the oracle $\mathcal{C}_{sk_1,pk_1,pk_2,h}$ under the multi-message security (2 messages) of El-Gamal encryption system (Theorem 2).*

Proof. Assume for the sake of contradiction that there exists an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the transformed ciphertext security of the encryption system such that

$$\Delta\big(IND_{0,tran}(\Pi, \mathcal{A}, \lambda), IND_{1,tran}(\Pi, \mathcal{A}, \lambda)\big) = \delta$$

where δ is non-negligible. We will construct an adversary $\mathcal{B}$ against the multi-message security of El-Gamal encryption system with non-negligible advantage.

$\mathcal{B}$ obtains the public key $pk = (g, g^y)$ and the public parameters *params* from the El-Gamal challenger. It chooses $x \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $pk_1 = (g, g^x)$ and $sk_1 = x$. It sets $pk_2 = (g, g^y)$ and implicitly defines sk_2 as y. It outputs $(pk_1, sk_1, pk_2, params)$ to the adversary $\mathcal{A}_1$.

It now chooses $h \xleftarrow{\$} \mathbb{G}$. Since $\mathcal{B}$ has access to both sk_1, pk_2 and h, it can simulate the re-encryption circuit perfectly. That is, it runs the program $Re - Enc_{1\to 2}$ with the constants sk_1, pk_1, pk_2, h hardcoded in the program description.

In the challenge phase, $\mathcal{A}_1$ outputs two messages m_0, m_1. $\mathcal{B}$ outputs $(h, 1)$ and (w, w') where $w, w' \xleftarrow{\$} \mathbb{G}$ (chosen independently) as the message sequences to the El-Gamal challenger. It obtains $c_2^* = (C_1^*, C_2^*)$ and $d_2^* = [D_1^*, D_2^*]$ as the challenge ciphertexts. It chooses $b \xleftarrow{\$} \{0, 1\}$, $r \xleftarrow{\$} \mathbb{Z}_p^*$ and outputs

$$(m_b \cdot g^r, (g^{H(h)})^r \cdot D_1^*, C_1^*, C_2^*, D_2^*)$$

It is easy to see that if c_2^* is an encryption of h and d_2^* is an encryption of 1, then the challenge ciphertext is distributed identically to $\mathcal{C}(\mathsf{Encrypt1}(m_1, pk_1,$
$params); r)$ where $r \xleftarrow{\$} \{0, 1\}^{3\lambda}$. Otherwise, each term in the challenge ciphertext is random (over the random choice of w, w', r as well as the random coins used for generating c_2^* and d_2^*) and independent of each other.

$\mathcal{A}_2$ outputs b'. If $b = b'$, then $\mathcal{B}$ outputs 1. Else, it outputs 0. Hence, the probability that $\mathcal{A}_2$ outputs $b' = b$ is $1/2 + \delta/2$ in the first case and is equal to $1/2$ in the second case. Hence, advantage of $\mathcal{B}$ against the El-Gamal challenger is given by $\delta/2$ which is non-negligible (a contradiction). □

5 Average-case Virtual Black Box Property

We show that obfuscator construction preserves functionality in Appendix D. We note that the polynomial slowdown property of our construction can be easily verified. It is interesting to note that the obfuscated circuit computes seven exponentiations whereas the original circuit computes eight exponentiations.

We now show that Obf satisfies the average-case virtual black box property.

Lemma 3. Obf *satisfies the average case secure virtual black-box property.*

Proof. The proof techniques used here are similar to that of Hohenberger et al. in [HRsV07] and the details follow.

Let $\mathcal{C} \leftarrow \mathsf{Samp}(1^\lambda)$ be a circuit chosen randomly from the set $\mathcal{C}_\lambda$ using the Samp algorithm. Let D be any distinguisher with oracle access to $\mathcal{C}$.

We first describe the simulator Sim which has oracle access to the circuit $\mathcal{C}$ and takes as input the security parameter in unary form and auxiliary information string denoted by z.

$\mathsf{Sim}^{\mathcal{O}(\mathcal{C})}$

Input: $1^\lambda, z$

1. Query the oracle $\mathcal{O}(\mathcal{C})$ with the special symbol *keys* and obtain pk_1 and pk_2.
2. Parse pk_2 as (g, g^y).
3. Choose $h \xleftarrow{\$} \mathbb{G}$, $v \xleftarrow{\$} \mathbb{Z}_p^*$ and compute $(Z_1', Z_2') = (h \cdot (g^y)^v, g^v)$. Choose Z_3' uniformly at random from $\mathbb{Z}_p^*$.
4. Construct a circuit $\mathcal{C}'$ implementing the program $\mathsf{Re-Enc}'_{1\to 2}$ with values $(pk_1, pk_2, Z_1', Z_2', Z_3')$ hardcoded in the program description.
5. Output the circuit description of $\mathcal{C}'$.

It remains to show that the output distribution of the simulator is computationally indistinguishable to the output distribution of Obf even to distinguishers having oracle access to $\mathcal{C}$.

We define two distributions $\mathsf{Nice}(D^{\mathcal{O}(\mathcal{C})}, \lambda, z)$ and $\mathsf{Junk}(D^{\mathcal{O}(\mathcal{C})}, \lambda, z)$ as follows:

$\mathsf{Nice}(D^{\mathcal{O}(\mathcal{C})}, \lambda, z)$

- $(p, g, \mathbb{G}, H) \leftarrow \mathsf{Setup}(1^\lambda)$
- Choose $x, y \xleftarrow{\$} \mathbb{Z}_p^*$.
- Set $pk_1 = (g, g^x)$ and $pk_2 = (g, g^y)$.
- Choose $h \xleftarrow{\$} \mathbb{G}$ and $v \xleftarrow{\$} \mathbb{Z}_p^*$.
- Compute $Z_1 = h \cdot (g^y)^v$, $Z_2 = g^v$ and $Z_3 = H(h)/x$.
- Output $D^{\mathcal{O}(\mathcal{C})}(pk_1, pk_2, Z_1, Z_2, Z_3, z)$.

$\mathsf{Junk}(D^{\mathcal{O}(\mathcal{C})}, \lambda, z)$

- $(p, g, \mathbb{G}, H) \leftarrow \mathsf{Setup}(1^\lambda)$
- Choose $x, y \xleftarrow{\$} \mathbb{Z}_p^*$.
- Set $pk_1 = (g, g^x)$ and $pk_2 = (g, g^y)$.
- Choose $h \xleftarrow{\$} \mathbb{G}$ and $v \xleftarrow{\$} \mathbb{Z}_p^*$.
- Compute $Z_1' = h \cdot (g^y)^v$, $Z_2' = g^v$ and $Z_3' \xleftarrow{\$} \mathbb{Z}_p^*$.
- Output $D^{\mathcal{O}(\mathcal{C})}(pk_1, pk_2, Z_1', Z_2', Z_3', z)$.

We first observe that for all $z \in \{0,1\}^{poly(\lambda)}$ and for all distinguishers D,

$$\{\mathcal{C} \to \mathsf{Samp}(1^\lambda); \mathcal{C}' \leftarrow \mathsf{Obf}(\mathcal{C}) : D^{\mathcal{O}(\mathcal{C})}(\mathcal{C}', z)\} \approx \mathsf{Nice}(D^{\mathcal{O}(\mathcal{C})}, \lambda, z)$$

$$\{\mathcal{C} \to \mathsf{Samp}(1^\lambda); \mathcal{C}' \leftarrow \mathsf{Sim}^{\mathcal{O}(\mathcal{C})}(1^\lambda, z) : D^{\mathcal{O}(\mathcal{C})}(\mathcal{C}', z)\} \approx \mathsf{Junk}(D^{\mathcal{O}(\mathcal{C})}, \lambda, z)$$

In order to show that Obf satisfies the average case virtual black box property it is enough to show that (from Lemma 1), for all PPT distinguishers D, there exists a negligible function $\mu(\cdot)$ such that for all $z \in \{0,1\}^{poly(\lambda)}$,

$$\Delta\big(\mathsf{Nice}(D^{\mathcal{O}(\mathcal{C})}, \lambda, z)\}, \mathsf{Junk}(D^{\mathcal{O}(\mathcal{C})}, \lambda, z)\big) \leq \mu(\lambda)$$

We show this through a series of propositions.

First, we consider two distributions which are similar to Nice and Junk except that they consider a "dummy" distinguisher D^* which outputs whatever is given as input.

Proposition 1. $\{\mathsf{Nice}(D^*, \lambda, z)\}_\lambda \stackrel{c}{\approx} \{\mathsf{Junk}(D^*, \lambda, z)\}_\lambda$

Proof. The proof for the this proposition follows directly from the proof of Lemma 2. We note that Hyb0 is identically distributed to $\mathsf{Junk}(D^*, \lambda, z)$ and Hyb2 is identically distributed to $\mathsf{Nice}(D^*, \lambda, z)$. Hence,

$$\mathsf{Nice}(D^*, \lambda, z) \stackrel{c}{\approx} \mathsf{Junk}(D^*, \lambda, z)$$

□

We now consider two more distributions which proceed as Nice and Junk except that they consider distinguishers $D^{\mathcal{O}(R)}$ where R is a probabilistic circuit which on any input $[C_1, C_2]$, first checks if C_1, C_2 belong to $\mathbb{G}$ and if yes, outputs $[A, B, C, D, E]$ where A, B, C, D, E are chosen uniformly and independently from $\mathbb{G}$. Otherwise, it outputs $\perp$.

Note that input to $D^{\mathcal{O}(R)}$ is identically distributed to $\mathsf{Nice}(D^*, \lambda, z)$ in $\mathsf{Nice}(D^{\mathcal{O}(R)}, \lambda, z)$ and its input is identically distributed to $\mathsf{Junk}(D^*, \lambda, z)$ in $\mathsf{Junk}(D^{\mathcal{O}(R)}, \lambda, z)$. The following proposition is a direct consequence of Proposition 1.

Proposition 2. *For all PPT distinguihsers D, there exists a negligible function $\mu(\cdot)$ such that for all $z \in \{0,1\}^{poly(\lambda)}$ and for all $\lambda \in \mathbb{N}$, we have*

$$\Delta\big(\mathsf{Nice}(D^{\mathcal{O}(R)}, \lambda, z), \mathsf{Junk}(D^{\mathcal{O}(R)}, \lambda, z)\big) \leq \mu(\lambda)$$

Proof. Assume for the sake of contradiction that there exists a distinguisher $D^{\mathcal{O}(R)}$ which can distinguish between $\mathsf{Nice}(D^*, \lambda, z)$ and $\mathsf{Junk}(D^*, \lambda, z)$ with non-negligible advantage. We construct an distinguisher between D' (without the oracle access to R) which distinguishes between $\mathsf{Nice}(D^*, \lambda, z)$ and $\mathsf{Junk}(D^*, \lambda, z)$ with the same advantage.

D' runs D internally by giving its own input as input to D. When D requests an oracle access to R, D' can simulate the responses on its own (It will choose five independent random elements from the group and return as the response for any oracle query after checking whether the input belongs to $\mathbb{G} \times \mathbb{G}$). D' finally outputs what D outputs.

It is easy to see that D' as the same distinguishing advantage that D has and hence we have arrived at a contradiction to Proposition 1. □

Consider any distinguisher D. Let us define,

$$\alpha(\lambda, z) = \Delta\big(\mathsf{Nice}(D^{\mathcal{O}(\mathcal{C})}, \lambda, z), \mathsf{Junk}(D^{\mathcal{O}(\mathcal{C})}, \lambda, z)\big)$$

$$\beta(\lambda, z) = \Delta\big(\mathsf{Nice}(D^{\mathcal{O}(R)}, \lambda, z), \mathsf{Junk}(D^{\mathcal{O}(R)}, \lambda, z)\big)$$

Let q_D be the number of oracle queries that D makes during its execution. Since D runs in polynomial time, q_D is polynomial in λ.

Proposition 3. *There exists an algorithm $\mathcal{B}$ against the multi-message ($2q_D$ messages) security of El-Gamal encryption scheme with an advantage $|\alpha(\lambda, z) - \beta(\lambda, z)|/2$.*

Proof. We prove the proposition by constructing an adversary $\mathcal{B}$ against the El-Gamal challenger with advantage $|\alpha(\lambda, z) - \beta(\lambda, z)|/2$.

$\mathcal{B}$ receives the public key g^y from the El-Gamal challenger. It chooses $x \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $pk_1 = (g, g^x)$ and $pk_2 = (g, g^y)$. It chooses two message vectors $\boldsymbol{M_0}$ and $\boldsymbol{M_1}$ each of length $2q_D$ as follows. It sets $\boldsymbol{M_0} = \{1, 1, ..., 1\}$ (of length $2q_D$) and $\boldsymbol{M_1} = \{m_1, m_2, ..., m_{2q_D}\}$ where $m_1, ..., m_{2q_D}$ are chosen uniformly and independently at random from $\mathbb{G}$. It then receives the challenge ciphertext vector $\boldsymbol{C^*} = \{(g^{r_1}, Q_1), (g^{r_2}, Q_2), \cdots, (g^{r_{q_D}}, Q_{2q_D})\}$ and auxiliary information z. Note that for all $i \in \{1, 2, ..2q_D\}$, r_i is a random element in $\mathbb{Z}_p^*$ and $Q_i = g^{yr_i}$ or an uniformly chosen element depending on whether $\boldsymbol{M_0}$ was encrypted or $\boldsymbol{M_1}$ was encrypted (due to the random choice of $m_1, ..., m_{2q_D}$).

$\mathcal{B}$ now uses D to determine whether the challenge ciphertext vector is an encryption of $\boldsymbol{M_0}$ or $\boldsymbol{M_1}$. It first generates the tuples which are distributed exactly as $\mathsf{Nice}(D^*, \lambda, z)$ and $\mathsf{Junk}(D^*, \lambda, z)$. It tosses a random coin c and runs D with input $\mathsf{Nice}(D^*, \lambda, z)$ if $c = 0$ and with input $\mathsf{Junk}(D^*, \lambda, z)$ if $c = 1$. $\mathcal{B}$ needs to answer the re-encryption oracle queries made by D. It uses the challenge ciphertext to answer those oracle queries. We show that if the challenge ciphertext is an encryption of $\boldsymbol{M_0}$, then the oracle responses given by $\mathcal{B}$ are identically distributed to the output of the re-encryption circuit $\mathcal{C}$. If the challenge ciphertext was an encryption of $\boldsymbol{M_1}$, we show that the oracle responses are identically distributed to the output of R. The exact details follow.

$\mathcal{B}$ chooses $h \xleftarrow{\$} \mathbb{G}$ and $v \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $Z_1 = h \cdot (g^y)^v$ and $Z_2 = g^v$. It then chooses $Z_3 = H(h)/x$ and $Z_3' \xleftarrow{\$} \mathbb{Z}_p^*$. It tosses a random coin and chooses $c \xleftarrow{\$} \{0, 1\}$. If $c = 0$, it runs $D^{\mathcal{O}(X)}(pk_1, pk_2, Z_1, Z_2, Z_3, z)$. Else it runs, $D^{\mathcal{O}(X)}(pk_1, pk_2, Z_1, Z_2, Z_3', z)$ where X is the circuit description of the program $\mathsf{Re-Enc}''_{1\to 2}$ described below. Note that if $c = 0$, input to D is identical to $\mathsf{Nice}(D^*, \lambda, z)$. Else, it is identical to $\mathsf{Junk}(D^*, \lambda, z)$.

When D makes i^{th} oracle query $[C_1, C_2]$, $\mathcal{B}$ runs the following program and returns the output of the program to D as the response.

$\mathsf{Re-Enc}''_{1\to 2}$

Constants: $pk_1, pk_2, Z_1, Z_2, Z_3$
Input: $[C_1, C_2], i$

1. Choose $r' \xleftarrow{\$} \mathbb{Z}_p^*$.
2. Compute $C_1' = C_1 \cdot g^{r'}$, $C_2' = C_2 \cdot (g^x)^{r'}$.
3. Compute $Z_1' = Z_1 \cdot (Q_i)$, $Z_2' = Z_2 \cdot (g^{r_i})$.
4. Compute $C_2'' = C_2'^{Z_3}$.
5. Compute $D_2 = C_2'' \cdot Q_{i+q_D}$
6. Output $[C_1', D_2, Z_1', Z_2', g^{r_{i+q_D}}]$.

D finally outputs its guess. Let c' denote the output of D. If $c = c'$, $\mathcal{B}$ outputs 1. Else, it outputs 0.

We now prove the following two claims regarding the output of $\mathcal{B}$.

Claim. If $\boldsymbol{M_0}$ was encrypted, the probability that $\mathcal{B}$ outputs 1 is given by $1/2 + \alpha(\lambda, z)/2$.

Proof. We claim that if $\boldsymbol{M_0}$ was encrypted, then $\mathcal{B}$ perfectly simulates $D^{\mathcal{O}(\mathcal{C})}\big(pk_1, pk_2, h \cdot (g^y)^r, g^r, H(h)/x, z\big)$ or $D^{\mathcal{O}(\mathcal{C})}\big(pk_1, pk_2, h \cdot (g^y)^r, g^r, Z_3', z\big)$ depending upon the bit c. We already noted that the input to D is identically distributed to $\mathsf{Nice}(D^*, \lambda, z)$ or $\mathsf{Junk}(D^*, \lambda, z)$. It is enough to show that X simulates the circuit $\mathcal{C}$ perfectly. Since $Q_i = (g^y)^{r_i}$ for all $i \in [1, 2q_D]$ and Z_1, Z_2, Z_3 are properly generated as per the Obf algorithm, the output of X is given by,

$$(m \cdot g^{r+r'}, (g^{r+r'})^{H(h)} \cdot (g^y)^{r_{i+q_D}}, h \cdot (g^y)^{v+r_i}, g^{v+r_i}, g^{r_{i+q_D}})$$

which is identically distributed as the output of the re-encryption circuit since r', r_i, r_{i+q_D} are chosen uniformly at random from $\mathbb{Z}_p^*$. Hence, the probability that $\mathcal{B}$ outputs 1 in this case is same as the probability that $D^{\mathcal{O}(\mathcal{C})}$ outputs $c = c'$ which is same as $1/2 + \alpha(\lambda, z)/2$. □

Claim. If $\boldsymbol{M_1}$ was encrypted, the probability that $\mathcal{B}$ outputs 1 is given by $1/2 + \beta(\lambda, z)/2$.

Proof. We already noted that the input to D are perfectly generated according to either $\mathsf{Nice}(D^*, \lambda, z)$ or $\mathsf{Junk}(D^*, \lambda, z)$. We claim that the response given by $\mathcal{B}$ are same as the one given by R. The output of $\mathcal{B}$ is given by

$$(m \cdot g^{r+r'}, (g^{r+r'})^{H(h)} \cdot Q_{i+q_D}, h \cdot (g^y)^v \cdot Q_i, g^{v+r_i}, g^{r_{i+q_D}})$$

Since Q_i and Q_{i+q_D} are uniformly chosen random elements in $\mathbb{G}$ if $\boldsymbol{M_1}$ was encrypted and r', r_i, r_{i+q_D} are chosen uniformly at random from $\mathbb{Z}_p^*$, we can easily see that all elements in the above distribution are random and independent for every invocation of the oracle.

Hence, in this case $\mathcal{B}$ perfectly simulates $D^{\mathcal{O}(R)}\big(pk_1, pk_2, h \cdot (g^y)^r, g^r, H(h)/x, z\big)$ or $D^{\mathcal{O}(R)}\big(pk_1, pk_2, h \cdot (g^y)^r, g^r, Z_3', z\big)$ depending on the bit c. Thus, the probability that $\mathcal{B}$ outputs 1 in this case is same as the probability that D^R outputs $c = c'$ which is given by $\beta(\lambda, z)/2 + 1/2$. □

Hence the advantage of $\mathcal{B}$ in the multi message security game of the El-Gamal Encryption scheme is given by $|\alpha(\lambda, z) - \beta(\lambda, z)|/2$. □

We know from Proposition 2 that $\beta(\lambda, z)$ is negligible. Hence from Proposition 3 we can infer that $\alpha(\lambda, z)$ is also negligible. Hence, Obf satisfies the average case secure virtual black box property and this concludes the proof of Lemma. □

6 Acknowledgement

The authors would like to thank the anonymous referees of PKC 2015 for pointing out a weakness in the security model for considered in the previous version of this work. AS would like to thank Dhinakaran Vinayagamurthy for insightful discussions on the topic.

References

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Advances in Cryptology-CRYPTO 2001*, pages 1–18. Springer, 2001.

[CCL+14] Nishanth Chandran, Melissa Chase, Feng-Hao Liu, Ryo Nishimaki, and Keita Xagawa. Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices. In *Public-Key Cryptography–PKC 2014*, pages 95–112. Springer, 2014.

[CCV12] Nishanth Chandran, Melissa Chase, and Vinod Vaikuntanathan. Functional re-encryption and collusion-resistant obfuscation. In *Theory of Cryptography*, pages 404–421. Springer, 2012.

[CH07] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 185–194. ACM, 2007.

[CWYD10] Sherman SM Chow, Jian Weng, Yanjiang Yang, and Robert H Deng. Efficient unidirectional proxy re-encryption. In *Progress in Cryptology–AFRICACRYPT 2010*, pages 316–332. Springer, 2010.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology*, pages 10–18. Springer, 1985.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[Had10] Satoshi Hada. Secure obfuscation for encrypted signatures. In *Advances in Cryptology–EUROCRYPT 2010*, pages 92–112. Springer, 2010.

[HRsV07] Susan Hohenberger, Guy N Rothblum, abhi shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. In *Theory of Cryptography*, pages 233–252. Springer, 2007.

[LV08] Benoît Libert and Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In *Public Key Cryptography–PKC 2008*, pages 360–379. Springer, 2008.

A Public Key Encryption

A PKE scheme consists of four algorithms: $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. Setup takes the unary encoding of the security parameter (1^λ) and outputs the set of public parameters denoted by $params$. The public parameters also describes the message space denoted by $\mathcal{M}$. KeyGen is a probabilistic algorithm which takes public parameters $params$ as input and outputs public key-secret key pair (pk, sk). The encryption algorithm Enc is a probabilistic algorithm that takes a message $m \in \mathcal{M}$, a public key pk and public parameters $params$ as input and outputs a ciphertext c. The decryption algorithm Dec is a deterministic algorithm which takes a ciphertext c, secret key sk and public parameters $params$ and outputs a message m. Dec outputs a special symbol denoted by $\perp$ when it is run on an invalid ciphertext. Any PKE system must satisfy the following correctness guarantee: for every $params \leftarrow \mathsf{Setup}(1^\lambda)$, every $(pk, sk) \leftarrow \mathsf{KeyGen}(params)$ and every $m \in \mathcal{M}$, we have $Pr[c \leftarrow \mathsf{Enc}(m, pk, params) : \mathsf{Dec}(c, sk, params) = m] = 1$.

We now define the single-message semantic security of an encryption scheme.

Definition 8. *Let $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE system. Let us define an experiment $\mathsf{IND}-\mathsf{CPA}_{one,b}\big(\Pi, \lambda, \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2), z\big)$ as follows:*

$\mathsf{IND}-\mathsf{CPA}_{one,b}\big(\Pi, \lambda, \mathcal{A}, z\big)$

1. *$params \leftarrow \mathsf{Setup}(1^\lambda)$.*
2. *$(pk, sk) \leftarrow \mathsf{KeyGen}(params)$.*
3. *$(m_0, m_1, state) \leftarrow \mathcal{A}_1(1^\lambda, params, pk)$.*
4. *$C^* \leftarrow \mathsf{Enc}(m_b, pk, params)$.*
5. *$b' \leftarrow \mathcal{A}_2(C^*, state, z)$.*
6. *Output b'*

*We say that Π is **single message secure** if for all non-uniform polynomial time adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mu(\cdot)$ such that for all $z \in \{0,1\}^{poly(\lambda)}$ and for all $\lambda \in \mathbb{N}$,*

$$\varDelta\Big(\mathsf{IND}-\mathsf{CPA}_{one,0}\big(\Pi, \lambda, \mathcal{A}, z\big), \mathsf{IND}-\mathsf{CPA}_{one,1}\big(\Pi, \lambda, \mathcal{A}, z\big)\Big) \leq \mu(\lambda)$$

We also define the standard multi-message security for an encryption scheme.

Definition 9. *Let $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE system. Let us define an experiment $\mathsf{IND}-\mathsf{CPA}_{many,b}\big(\Pi, \lambda, \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2), q(\cdot), z\big)$ as follows:*

$\mathsf{IND}-\mathsf{CPA}_{many,b}\big(\Pi, \lambda, \mathcal{A}, q(\cdot), z\big)$

1. *$params \leftarrow \mathsf{Setup}(1^\lambda)$.*
2. *$(pk, sk) \leftarrow \mathsf{KeyGen}(params)$.*
3. *$(\boldsymbol{M_0}, \boldsymbol{M_1}, state) \leftarrow \mathcal{A}_1(1^\lambda, params, pk, q(\cdot))$ where $|\boldsymbol{M_0}| = |\boldsymbol{M_1}| = q(\lambda)$.*
4. *$\boldsymbol{C^*} \leftarrow \{\mathsf{Enc}(m, pk, params)\}_{m \in \boldsymbol{M_b}}$.*
5. *$b' \leftarrow \mathcal{A}_2(\boldsymbol{C^*}, state, z)$.*
6. *Output b'*

*We say that Π is **multi message secure** if for all polynomials $q(\cdot)$, for all non-uniform polynomial time adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mu(\cdot)$ such that for all $z \in \{0,1\}^{poly(\lambda)}$ and for all $\lambda \in \mathbb{N}$,*

$$\varDelta\Big(\mathsf{IND}-\mathsf{CPA}_{many,0}\big(\Pi, \lambda, \mathcal{A}, q(\cdot), z\big), \mathsf{IND}-\mathsf{CPA}_{many,1}\big(\Pi, \lambda, \mathcal{A}, q(\cdot), z\big)\Big) \leq \mu(\lambda)$$

B El-Gamal Encryption System

B.1 El-Gamal Encryption System

El-Gamal Encryption System

- $\mathsf{Setup}(1^\lambda)$: Let $(p, \mathbb{G}, g) \leftarrow \mathsf{Gen}(1^\lambda)$. Output $params = (\mathbb{G}, p, g)$ as the public parameters with the message space $\mathcal{M} = \mathbb{G}$.
- $\mathsf{KeyGen}(params)$: Choose $x \xleftarrow{\$} \mathbb{Z}_p^*$ and set public key $pk = g^x$ and the secret key $sk = x$.
- $\mathsf{Enc}(m, pk, params)$: Parse the public key as $pk = g^x$. Choose $r \xleftarrow{\$} \mathbb{Z}_p^*$ and output the ciphertext $c = (m \cdot (g^x)^r, g^r)$.
- $\mathsf{Dec}(c, sk, params)$: Parse the secret key $sk = x$ and $c = (C_1, C_2)$. Output the message $m = C_1 \cdot (C_2^x)^{-1}$.

B.2 Variant of El-Gamal Encryption System

El-Gamal Variant

- $\mathsf{Setup}(1^\lambda)$: Let $(p, \mathbb{G}, g) \leftarrow \mathsf{Gen}(1^\lambda)$. Output $params = (\mathbb{G}, p, g)$ as the public parameters with the message space $\mathcal{M} = \mathbb{G}$.
- $\mathsf{KeyGen}(params)$: Choose $x \xleftarrow{\$} \mathbb{Z}_p^*$ and set public key $pk = g^x$ and the secret key $sk = x$.
- $\mathsf{Enc}(m, pk, params)$: Parse the public key as $pk = g^x$. Choose $r \xleftarrow{\$} \mathbb{Z}_p^*$ and output the ciphertext $c = ((g^x)^r, g^r \cdot m)$.
- $\mathsf{Dec}(c, sk, params)$: Parse the secret key $sk = x$ and $c = (C_1, C_2)$. Output the message $m = C_2 \cdot (C_1^{1/x})^{-1}$.

It is easy to see that El-Gamal encryption variant is both single and multi message secure under the DDH-assumption.

C Correctness

We observe that correctness of $\mathsf{Decrypt1}$ algorithm directly follows from the correctness of El-Gamal encryption scheme.

We now show the correctness of $\mathsf{Decrypt2}$ algorithm. The input to $\mathsf{Decrypt2}$ algorithm is given by $[C_1', C_2', C_3', C_4', C_5'] = [m \cdot g^r, (g^{H(h)})^r \cdot (g^y)^s, h \cdot (g^y)^v, g^v, g^s]$ and the secret key y. It first computes $C_3' \cdot ({C_4'}^y)^{-1} = h \cdot (g^y)^v \cdot ((g^v)^y)^{-1} = h$ and $C_2'' = C_2' \cdot ((C_5')^y)^{-1} = (g^{H(h)})^r \cdot (g^y)^s \cdot ((g^s)^y)^{-1} = (g^{H(h)})^r$. It then outputs $(C_1') \cdot ((C_2'')^{1/(H(h))})^{-1} = (m \cdot g^r) \cdot (((g^{H(h)})^r)^{1/(H(h))})^{-1} = m$.

D Preserving Functionality

The lemma stated below shows that the $\mathsf{Obf}(\mathcal{C})(c; \cdot)$ is identically distributed to $\mathcal{C}(c, \cdot)$ where $c \leftarrow \mathsf{Encrypt1}(m, pk_1, params)$.

Lemma 4. *Let $\mathcal{C}$ be a any circuit in $\mathcal{C}_\lambda$. Then, $\mathsf{Obf}(\mathcal{C})(c; \cdot)$ and $\mathcal{C}(c, \cdot)$ are identically distributed where $c \leftarrow \mathsf{Encrypt1}(m, pk_1, params)$.*

Proof. We prove this by considering the output distributions of $\mathcal{C}$ and $\mathsf{Obf}(\mathcal{C})$ on an input ciphertext $c = (m \cdot g^r, (g^x)^r)$.

Let us first consider the distribution $C(c)$. It is given by,

$$(m \cdot g^{r'}, (g^{r'})^{H(h)} \cdot (g^y)^s, h \cdot (g^y)^v, g^v, g^s)$$

where r', v, s are independently chosen random values from $\mathbb{Z}_p^*$.

When the same input is fed into $Obf(C)$ the output is given by,

$$(m \cdot g^{r+r'}, (g^{r+r'})^{H(h)} \cdot (g^y)^s, h \cdot (g^y)^{v+v'}, g^{v+v'}, g^s)$$

where r', v', s are uniformly and independently chosen values from $\mathbb{Z}_p^*$.

Let us denote $r + r'$ as $\overline{r}$ and $v + v'$ as $\overline{v}$. We note that $\overline{r}$ and $\overline{v}$ are uniformly distributed and are independent since r' and v' are independently chosen random values. Rewriting the above tuple we get,

$$(m \cdot g^{\overline{r}}, (g^{\overline{r}})^{H(h)} \cdot (g^y)^s, h \cdot (g^y)^{\overline{v}}, g^{\overline{v}}, g^s)$$

which is identically distributed as the output of C since $\overline{r}, \overline{v}, s$ are uniformly and independently distributed in $\mathbb{Z}_p^*$. $\square$