Efficiently Obfuscating Re-Encryption Program under DDH Assumption

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

[1] University of California, Berkeley
[2] Indian Institute of Technology, Madras

Abstract. A re-encryption program (or a circuit) transforms 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 may have to perform a polynomial number of pairing operations in the worst case. Moreover, the proof of security relies on strong assumptions. On the other hand, the construction given by Chandran et al. relies on standard assumptions on lattices but could only satisfy a relaxed notion of correctness.
In this work we propose a simple and efficient obfuscator for the re-encryption functionality which doesn't suffer from *any* of the above mentioned drawbacks. In particular, our construction satisfies the strongest notion of correctness, supports encryption of messages from an exponential sized domain and relies on the standard DDH-assumption. 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 [Had00,BGI+01] is a compiler that takes as input a description of a program and outputs a description of an equivalent program which is "unintelligible." The notion of program obfuscation was formalized in the

* Work partially done while author was a student at IIT-Madras. Email: akshayram1993@gmail.com
** Email: prangan55@gmail.com

seminal work of Barak et al. [BGI+01]. The strongest notion of obfuscation considered in their work is known as the *predicate black-box obfuscation* (also known as virtual black-box). 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. Barak et al. showed that a *general* program obfuscator satisfying the predicate black-box property does not exist. In spite of the impossibility result, program obfuscators satisfying the strong security notion have been constructed for simple function families in [CD08],[Wee05],[CRV10].

Average-case Secure Obfuscation. Hohenberger, Rothblum, shelat and Vaikuntanathan [HRsV07] noted that predicate black-box property [BGI+01] does not 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 discussed a scenario where having access to the obfuscated program could compromise the security of the larger cryptographic system even if the obfuscator satisfied the predicate black-box property.

To address the above issue, Hohenberger et al. proposed a new definition of obfuscation which they termed as *Average-case Secure Obfuscation.* This definition guarantees that any adversary against a cryptographic scheme having access to an obfuscated program (where the program itself is sampled from an appropriate distribution) can be transformed into an adversary against the cryptographic scheme with only black box access to the sampled program assuming the scheme has *distinguishable attack property.* Informally, 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. Hohenberger et al. showed that several natural cryptographic functionalities like semantically secure encryption and re-encryption have this property.

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.

1.1 Prior Work

Hohenberger et al. [HRsV07] designed an average case secure obfuscator for the re-encryption functionality under Decision Linear and a strong variant of 3-party Decisional Diffie-Hellman assumptions. The construction has a couple of drawbacks which we now elaborate. The first drawback is that it could only *support a message space of polynomial size.* Their decryption algorithm does an exhaustive search on the message space by computing a pairing operation on each message and tests the output of the pairing against a specific value. Thus, it has to *compute a polynomial number of pairing operations in the worst case.* Moreover, the security of their construction is based on a strong assumption namely, Strong 3-party DDH.

Remark 1. We note that it is possible to 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 it is desirable 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 assuming interactability of certain lattice problems. But their construction could only satisfy relaxed notions of correctness. In particular, they considered three relaxations of the correctness property. 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 that arises from the prior work is:

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

1.2 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. We 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 - re-encryption sustem system under the standard DDH assumption. All our proofs are 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 the 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 the re-encryption program in the average-case obfuscation paradigm, we must first 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 [3] under pk_1 or a first level ciphertext under pk_2 or a second level ciphertext [4] under pk_2. 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 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 $c \leftarrow \mathsf{Encrypt}(m, pk_1)$, 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 (e.g. the standard El-Gamal encryption).

[3] 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.

[4] 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

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. The goal of the adversary in both the games is to distinguish between encryptions of messages from encryptions of junk values. Additionally, we require the encryption - re-encryption system to satisfy a special property called as *statistical independence.* Statistical independence requires that the output distribution of the re-encryption program (i.e. the distribution of the second level ciphertext) to be *statistically independent* of sk_1. 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 ciphertext to obtain the hidden message. We also stress that statistical independence property guarantees that the second level ciphertext cannot "leak" any information (in an information theoretic sense) regarding sk_1. This in particular, disallows other contrived examples which may reveal the secret key sk_1 to Bob but possibly is still transformed ciphertext secure.[5]

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.[6] 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.

[5] For example, we could consider the output of the re-encryption functionality to be $(\mathsf{Encrypt}(C, pk_2)||\mathsf{Encrypt}(sk_1, pk_2))$ where C is the input ciphertext. This is transformed ciphertext secure. We thank an anonymous CT-RSA 2016 referee for pointing out this deficiency in the security model. However, the output of the re-encryption functionally is dependent on sk_1.

[6] 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

1.3 Related Work

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 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 the *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. A (non-exhaustive) list of proxy re-encryption schemes under different notions of security can be found in [BBS98], [AFGH06], [CH07], [LV08] and [CWYD10]. But as noted in the work of Hohenberger et al. [HRsV07], the above works cannot be considered as an obfuscation of the re-encryption circuit. In particular, obfuscation of re-encryption circuit guarantees that no *non-black-box* information about the re-encryption circuit is "leaked" to the proxy. On the other hand, it is not directly evident if such guarantees can be given from proxy re-encryption systems as it may be possible that the re-key could leak some non-black-box information. We leave the possibility of strengthening the security definition of proxy re-encryption to guarantee that no non-black-box information is leaked as an interesting open problem.

Indistinguishability Obfuscation. Since the strongest notion of program obfuscation (namely, predicate black-box obfuscation) was shown to be impossible, Barak et al. [BGI$^+$01] proposed a weakened notion of obfuscation called as *indistinguishability obfuscation* or iO. Indistinguishability obfuscation guarantees that for any two functionally equivalent circuits having the same size, obfuscations of the circuits are indistinguishable. The first candidate construction of iO was given in the recent breakthrough work of Garg et al. [GGH$^+$13]. Subsequently, several cryptographic primitives like functional encryption [GGH$^+$13], deniable encryption [SW14], non-interactive key exchange without a trusted setup [BZ14], two-round multiparty computation protocols [GGHR14] and trapdoor permutations [BPW16] (to name a few) were constructed from iO and other assumptions like one-way functions. We note that indistinguishability guarantee provided by iO is strictly weaker than the security guarantee needed in this work. In addition, our goal is to obfuscate a specific functionality namely, the re-encryption functionality.

1.4 Organization

We give the standard definitions in Section 2 and recall the definition of average-case secure obfuscation in Section 2.1. We give the construction of encryption - re-encryption functionality and the construction of the obfuscator for re-encryption circuit in Section 3. We discuss the new (strengthened) black-box model for proving security of the encryption - re-encryption program in Section 4 and the prove the security of our construction in the same model. Finally, in Section 5 we show that the obfuscator described in Section 3 is average-case secure.

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 $x \overset{\$}{\leftarrow} X$ 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 $\mathcal{A}$. 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. PPT machines refer to Probabilistic Polynomial Time Turing 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 assume familiarity with the notion of computational indistinguishability and statistical distance (a.k.a. variation distance) and skip the standard definitions. 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,*

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

Proof. The lemma directly follows from the following observation that, $\big|Pr[b \leftarrow D(X_n, z) : b = 1] - Pr[b \leftarrow D(Y_n, z) : b = 1]\big|$ and $\big|Pr[b \leftarrow D(X_n, z) : b = 0] - Pr[b \leftarrow D(Y_n, z) : b = 0]\big|$ 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. $\square$

This lemma implies that $\{X_n\}_n \overset{c}{\approx} \{Y_n\}_n$ 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 1 (DDH assumption). *The DDH assumption states that the following distribution ensembles are computationally indistinguishable:*

$$\{(p, \mathbb{G}, g) \leftarrow \mathsf{Gen}(1^\lambda); a, b \xleftarrow{\$} \mathbb{Z}_p^* : (g, g^a, g^b, g^{ab})\}_\lambda \overset{c}{\approx}$$
$$\{(p, \mathbb{G}, g) \leftarrow \mathsf{Gen}(1^\lambda); a, b, c \xleftarrow{\$} \mathbb{Z}_p^* : (g, g^a, g^b, g^c)\}_\lambda$$

We recall the syntax and security notion (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 recall the theorem regarding the multi-message security of El-Gamal encryption system and its variant.

Theorem 1 (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 assume familiarity with of the concept of Pseudo Random Generator (PRG) and refer the reader to [Gol01] for a formal definition.

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 $\mathcal{O}$. 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 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 2 ([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* $z \in \{0,1\}^{poly(\lambda)}$*:*
$$|Pr[\mathcal{C} \leftarrow \mathsf{Samp}(1^\lambda); \mathcal{C}' \leftarrow \mathsf{Obf}(\mathcal{C}); b \leftarrow D^{\mathcal{O}(\mathcal{C})}(\mathcal{C}', z) : b = 1] -$$
$$Pr[\mathcal{C} \leftarrow \mathsf{Samp}(1^\lambda); \mathcal{C}' \leftarrow \mathsf{Sim}^{\mathcal{O}(\mathcal{C})}(1^\lambda, z); b \leftarrow D^{\mathcal{O}(\mathcal{C})}(\mathcal{C}', z) : b = 1]| \leq \mathsf{neg}(\lambda)$$

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.

New Encryption Scheme. The new encryption system under consideration is same as the El-Gamal system variant described in Appendix B 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 standard definition of pseudo random generator 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.

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 $\mathsf{Re} - \mathsf{Enc}_{1\to 2}$) is described below.

```
                    $\mathsf{Re} - \mathsf{Enc}_{1\to 2}$
Input: $c_1 = [C_1, C_2]$ or special symbol denoted by $\mathsf{keys}$[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 $\mathcal{C}_\lambda$ proceeds by choosing $(p, \mathbb{G}, g, H) \leftarrow \mathsf{Setup}(1^\lambda)$.

It then samples $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $h \xleftarrow{\$} \mathbb{G}$. It finally outputs 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. We note that decrypting the circuit output is straightforward and describe the actual decryption procedure in Appendix C.

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.

```
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 $\mathsf{Re-Enc}'_{1\to2}$
   described below with $pk_1, pk_2, Z_1, Z_2, Z_3$ as the constants in the program.
```

```
$\mathsf{Re-Enc}'_{1\to2}$
Input: $c_1 = [C_1, C_2]$ or special symbol denoted by $\mathsf{keys}$.
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 $\mathsf{Re-Enc}'_{1\to2}$. 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 1.2, we modify the security model given in [HRsV07] as follows.

4.1 Security Model

Let $\mathcal{C} \leftarrow \mathsf{Samp}(1^\lambda)$ [7] 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 3. *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. *$params \leftarrow \mathsf{Setup}(1^\lambda)$. $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(params)$ and $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}(params)$. Choose $h \xleftarrow{\$} \mathbb{G}$. Set $\mathcal{C} = \mathcal{C}_{sk_1,pk_1,pk_2,h}$ [a].*
2. *$(m_0, m_1, state) \leftarrow \mathcal{A}_1^{\mathcal{O}(\mathcal{C})}(pk_1, pk_2, params)$.*
3. *$C^* \leftarrow \mathsf{Encrypt1}(m_b, pk_i, params)$.*
4. *$b' \leftarrow \mathcal{A}_2^{\mathcal{O}(\mathcal{C})}(C^*, state)$. Output b'*

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

The scheme $\mathit{\Pi}$ 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 $\lambda \in \mathbb{N}$,*

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

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

Definition 4. *Let $\mathit{\Pi}$ be an encryption scheme and let $IND_{b,tran}(\mathit{\Pi}, \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2), \lambda)$ where $b \in \{0,1\}$ denote the following experiment:*

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

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

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

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

Statistical Independence. Let us consider the following experiment.

$\mathsf{Stat}(\Pi, \lambda, \mathsf{m})$

1. *params* $\leftarrow \mathsf{Setup}(1^\lambda)$. $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(params)$ and $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}(params)$. Choose $h \xleftarrow{\$} \mathbb{G}$. Set $\mathcal{C} = \mathcal{C}_{sk_1,pk_1,pk_2,h}$.
2. $\mathsf{rand} \xleftarrow{\$} \{0,1\}^{3\lambda}$. Compute $C^* \leftarrow \mathcal{C}(\mathsf{Encrypt1}(m, pk_1, params); \mathsf{rand})$.
3. Output C^*

We require the output of $\mathsf{Stat}(\Pi, \lambda, \mathsf{m})$ to be statistically independent of sk_1.

4.2 Security Proof

We now show that the New Encryption Scheme is *original ciphertext secure* (in Theorem 4), *transformed ciphertext secure* (in Theorem 5) and has statistical independence property (in Lemma 2).

Theorem 2. *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 show the proof of Theorem in Appendix E.

We now show the transformed ciphertext security of our construction.

Theorem 3. *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 1).*

Proof. We give the proof in Appendix F.

We note that the statistical independence property of the re-encryption functionality directly follows from inspection of the output distribution of the re-encryption circuit. We record the following lemma.

Lemma 2. *The output distribution of $\mathcal{C}_{sk_1,pk_1,pk_2,h}$ where $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}$ $(params)$, $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}(params)$ and $h \xleftarrow{\$} \mathbb{G}$ is statistically independent of sk_1.*

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

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

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

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 in Appendix G.

□

Since Obf satisfies the three requirements given in Definition 2, we conclude that Obf is an average-case secure obfuscator.

5.1 Acknowledgements

We would like to thank the anonymous reviewers of PKC, 2015 for insightful reviews.

References

[AFGH06] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, 2006.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *Advances in Cryptology-EUROCRYPT'98*, pages 127–144. Springer, 1998.

[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.

[BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 474–502, 2016.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 480–499, 2014.

[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.

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Obfuscating point functions with multibit output. In *Advances in Cryptology–EUROCRYPT 2008*, pages 489–508. Springer, 2008.

[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.

[CRV10] Ran Canetti, Guy N Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In *Theory of Cryptography*, pages 72–89. Springer, 2010.

[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.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 40–49. IEEE, 2013.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 74–94, 2014.

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

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, pages 443–457, 2000.

[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.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484, 2014.

[Wee05] Hoeteck Wee. On obfuscating point functions. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 523–532. ACM, 2005.

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

Definition 5. *Let $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE system. Let us define an experiment*
$\mathsf{IND-CPA}_{many,b}\big(\Pi, \lambda, \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2), q(\cdot), z\big)$ *as follows:*

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

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 $\lambda \in \mathbb{N}$,*

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

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

The output of $\mathsf{Re-Enc}_{1\to 2}$ 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 $m = (C_1') \cdot ((C_2'')^{1/(H(h))})^{-1}$.
```

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 $\mathbb{Z}_p^*$. $\square$

E Original Ciphertext Security

Theorem 4. *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 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 5.

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

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

$$\mathsf{Hyb0} \approx \left\{\begin{array}{l}(p, g, \mathbb{G}, H) \leftarrow \mathsf{Setup}(1^\lambda);\\ x, y \xleftarrow{\$} \mathbb{Z}_p^*;\\ pk_1 = (g, g^x);\\ pk_2 = (g, g^y);\\ h \xleftarrow{\$} \mathbb{G};\\ Z_3', v \xleftarrow{\$} \mathbb{Z}_p^* :\\ \big(pk_1, pk_2, h \cdot (g^y)^v, g^v, Z_3'\big)\end{array}\right\}_\lambda \quad \mathsf{Hyb1} \approx \left\{\begin{array}{l}(p, g, \mathbb{G}, H) \leftarrow \mathsf{Setup}(1^\lambda);\\ x, y \xleftarrow{\$} \mathbb{Z}_p^*;\\ pk_1 = (g, g^x);\\ pk_2 = (g, g^y);\\ h, h' \xleftarrow{\$} \mathbb{G};\\ v \xleftarrow{\$} \mathbb{Z}_p^* :\\ \big(pk_1, pk_2, h \cdot (g^y)^v, g^v, H(h')/x\big)\end{array}\right\}_\lambda$$

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

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 $\mathsf{Re}-\mathsf{Enc}'_{1\to2}(pk_1, pk_2, Z_1, Z_2, H(h)/x)$ preserves the same functionality as that of the original circuit (proved in Appendix 4). Note that the functionality of $\mathsf{Re}-\mathsf{Enc}'_{1\to2}$ 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:

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

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 $x \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and sets $pk_1 = (g, g^x)$ and $pk_2 = (g, g^y)$. $\mathcal{B}$ chooses $h \overset{\$}{\leftarrow} \mathbb{G}$.

$\mathcal{B}$ 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 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.

$\square$

F Transformed Ciphertext Security

Theorem 5. *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 1).*

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). $\square$

G Proof of Lemma 3

We show that for all PPT distinguishers D, there exists a negligible function $\mu(\cdot)$ such that for all $z \in \{0, 1\}^{poly(\lambda)}$,

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

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

Proof. The proof for the this proposition follows directly from the proof of Lemma 5 which appears in Appendix 4.2. We note that $\mathsf{Hyb0}$ is identically distributed to $\mathsf{Junk}(D^*, \lambda, z)$ and $\mathsf{Hyb2}$ is identically distributed to $\mathsf{Nice}(D^*, \lambda, z)$. Hence,

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

□

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 $\boldsymbol{C^*} = \{(g^{r_1}, Q_1), (g^{r_2}, Q_2), ...,$
$(g^{r_{q_D}}, Q_{2q_D})\}$ 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.