J. Cryptol. (2011) 24: 694–719
DOI: 10.1007/s00145-010-9077-7

Journal of
CRYPTOLOGY

Securely Obfuscating Re-Encryption*

Susan Hohenberger†
Johns Hopkins University, Baltimore, MD 21218, USA
susan@cs.jhu.edu

Guy N. Rothblum‡
Princeton University, Princeton, NJ 08544, USA
rothblum@alum.mit.edu

Abhi Shelat§
University of Virginia, Charlottesville, VA 22903, USA
abhi@virginia.edu

Vinod Vaikuntanathan¶
Microsoft Research Redmond, Redmond, WA 98052, USA
vinodv@alum.mit.edu

Communicated by Cynthia Dwork

Received 1 August 2007
Online publication 8 September 2010

Abstract. We present a positive obfuscation result for a traditional cryptographic functionality. This positive result stands in contrast to well-known impossibility results (Barak et al. in Advances in Cryptology—CRYPTO'01, 2002), for *general obfuscation* and recent impossibility and implausibility (Goldwasser and Kalai in 46th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 553–562, 2005) results for obfuscation of many cryptographic functionalities.

Whereas other positive obfuscation results in the standard model apply to very simple point functions (Canetti in Advances in Cryptology—CRYPTO'97, 1997; Wee in 37th ACM Symposium on Theory of Computing (STOC), pp. 523–532, 2005), our obfuscation result applies to the significantly more complex and widely-used *re-*

* This paper was solicited by the Editors-in-Chief as one of the best papers from TCC 2007, based on the recommendation of the program committee.

† Research partially performed at IBM Zurich Research Laboratory, Switzerland and partially supported by NSF grant CNS-0716142 and a Microsoft New Faculty Fellowship.

‡ Research partially performed while at MIT, and supported by NSF Grants CCF-0635297, CCF-0832797 and by a Computing Innovation Fellowship.

§ Research partially performed at IBM Zurich Research Laboratory, Switzerland and partially supported by NSF grant CNS-0845811.

¶ Research performed while at MIT, and supported by NSF grant CNS-0430450 and NSF grant CFF-0635297.

encryption functionality. This functionality takes a ciphertext for message m encrypted under Alice's public key and transforms it into a ciphertext for the same message m under Bob's public key.

To overcome impossibility results and to make our results meaningful for cryptographic functionalities, our scheme satisfies a definition of obfuscation which incorporates more security-aware provisions.

Key words. Obfuscation, Re-encryption.

1. Introduction

A recent line of research in theoretical cryptography aims to understand whether it is possible to *obfuscate* programs so that a program's code becomes unintelligible while its functionality remains unchanged. A general method for obfuscating programs would lead to the solution of many open problems in cryptography.

Hada [14] was the first to explicitly consider the problem of obfuscation, and gave negative results for obfuscating pseudo-random functions under a strong definition. Later, Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan and Yang [2] show that for many notions of obfuscation, a general program obfuscator does not exist—i.e., they exhibit a class of circuits which cannot be obfuscated. A subsequent work of Goldwasser and Kalai [12] shows the impossibility and implausibility of obfuscating more natural functionalities.

In spite of these negative results for general-purpose obfuscation, there are a few positive obfuscation results for simple functionalities such as point functions. A point function I_x returns 1 on input x and 0 on all other inputs. Canetti [7] shows that under a very strong Diffie–Hellman assumption point functions can be obfuscated. Further work of Canetti, Micciancio and Reingold [8], Wee [20] and Dodis and Smith [10] relaxes the assumptions required for obfuscation and considers other (related) functionalities. Despite these positive results, obfuscators for traditional cryptographic functionalities (such as those that deal with encryption) have remained elusive.

Our Results In this work, we present an obfuscator for a difficult cryptographic task:

Main Theorem (Informal). *Under standard bilinear complexity assumptions, there exists a secure encryption scheme and an efficient program obfuscator for a family of circuits implementing re-encryption.*

A *re-encryption program* for Alice and Bob takes a ciphertext for a message m encrypted under Alice's public key, and transforms it into a ciphertext (possibly of a different form) for the same message m under Bob's public key. Re-encryption programs have many practical applications such as the iTunes DRM system (albeit, with symmetric keys [18]), secure distributed file servers [1] and secure email forwarding.

The straightforward method to implement re-encryption is to write a program P which decrypts the input ciphertext using Alice's secret key and then encrypts the resulting message with Bob's public key. When P is run by Alice, this approach is reasonable.

In the practical applications noted above, however, the re-encryption program is executed by a third party. When this is the case, the straightforward implementation has

serious security concerns since P's code may reveal Alice's secret key to the third party. A better solution is to use an obfuscator for the re-encryption program P. That is, the third party is given an obfuscated program P' which computes the same function as P, but does not reveal anything more than the third party could learn from interaction with a black-box oracle that computes P.

As we discuss later in Sect. 1.2, several re-encryption schemes have been proposed before [1,3,4,9], but none of these prior works satisfy the strong obfuscation requirement informally stated above. Our main technical contribution is the construction of a novel re-encryption scheme which meets this strong notion while remaining surprisingly practical. In our construction, ciphertexts that are re-encrypted from Alice to Bob change in format and cannot be further re-encrypted from Bob to Carol. This may be a limitation in some scenarios, but it is nonetheless sufficient for several practical applications.

The obfuscator for re-encryption presented in this work sidesteps impossibility results by considering the average-case security of randomized functionalities. Its security is proven under an application-oriented definition of obfuscation based on a suggestion of Pass [17]. This definition is more meaningful for cryptographic applications than previous definitions of obfuscation.

1.1. *Notion of Average-Case Secure Obfuscation*

The work of [2] models an obfuscator as a compiler which takes a program (i.e., boolean circuit) P and produces an equivalent program that satisfies the *predicate black-box* property: any *predicate* that is computable from the obfuscated program should also be computable from black-box access to the program (see Definition 2.1).

Composable Obfuscation Unfortunately, the predicate definition [2] and subsequent work does not provide a meaningful security guarantee when the obfuscated program is used as part of a larger cryptographic system.[1] Intuitively, while the predicate black-box property gives a quantifiable guarantee that *some* information (namely, predicates) about the program is hidden by the obfuscated circuit, it does not guarantee that other "non-black-box information" does not leak. Moreover, this leaked information might compromise the security of a cryptographic scheme which uses the obfuscated circuit. For instance, it is completely possible that an obfuscated program that produces digital signatures (e.g., a program for "signature delegation") both meets the predicate black-box definition and is unforgeable under black-box access to a signature oracle, yet allows an adversary who has the obfuscated program code to forge a signature.[2]

[1] See also the (independent) discussion in Hofheinz, Malone-Lee and Stam [15].

[2] We do not know of any obfuscators for digital signatures satisfying the predicate black-box definition. Suppose one exists and that we alter it slightly so that into the description of each obfuscated circuit is wired a random message m together with its signature σ. The signature scheme is unforgeable under black-box access to a signature oracle, but any adversary given the obfuscated circuit can clearly win the forgery game by outputting (m, σ). It seems plausible, however, that this altered obfuscator would satisfy the predicate black-box definition, since the simulator may also choose a random message and obtain a signature on it, while the predicate's input is independent of the randomness used in creating the obfuscation and therefore the pair (m, σ).

Since many potential applications of obfuscation use obfuscated circuits in larger cryptographic schemes, the definition of obfuscation *should* guarantee that the security of cryptographic schemes is preserved in the following sense:

> *If a cryptographic scheme is "secure" when the adversary is given black-box access to a program, then it remains "secure" when the adversary is given an obfuscated version of the program.*

Such definitions were suggested independently by Pass [17] and (a slightly relaxed variant) by Hofheinz, Malone-Lee and Stam [15]. We prove the security of our construction under a variant of these definitions. Informally, the guarantee we seek requires that if there exists an adversary that uses the obfuscated program code to break the security of a cryptographic scheme, then there also exists a *black-box adversary* that breaks the scheme with similar probability using only black-box access to the program. Thus, if a scheme is secure against adversaries with only black-box access, then it is also secure against adversaries with access to obfuscated programs which implement the program. The definition we use in this work gives the above guarantee for any cryptographic scheme where a distinguisher with black-box access to the obfuscated functionality can simulate the security game with the adversary. Semantically secure encryption and re-encryption (defined in Sect. 4.3) are examples of such schemes.

Obfuscating Probabilistic Circuits It is not hard to see that *deterministic* functionalities that are not (approximately) learnable cannot be obfuscated under the composable definitions of obfuscation (see Sect. 2.2). In fact, composable definitions of obfuscation from [2,20] and [15,17] are only achievable for learnable deterministic functions. An important conceptual contribution of this work is showing that these impossibility results disappear when considering obfuscation of *probabilistic* functionalities—indeed, the functionality we obfuscate in this paper is a complex non-learnable probabilistic functionality. In addition, obfuscation of probabilistic circuits is important because most interesting cryptographic functionalities are probabilistic. See Sect. 2.2 for a discussion.

Using Obfuscation to Build Cryptographic Schemes The construction of an obfuscated re-encryption scheme suggests that (secure) obfuscation can play an important role in the design of cryptographic schemes. With secure obfuscation, the design of such schemes proceeds in two stages:

1. Specify the functionality of a program (or program family), and prove security of the cryptographic scheme against an adversary given *black-box* access to the program.
2. Construct a secure obfuscator for the program (or program family).

The combination of these two steps guarantees security of the scheme against an adversary that has full access to the obfuscated program. For our scheme, we show step (1) in Theorem 4.5 and step (2) in Theorem 5.1. Note that the security of the obfuscator (together with the scheme's black-box security) could imply many desirable security properties at once. In particular, the obfuscated programs are as secure against any "attack" (i.e., a probabilistic experiment which can be run in polynomial time with oracle access to the functionality) as a black-box would be. Thus the re-encryption scheme

presented in this paper may prove to be more secure than known re-encryption schemes which only have the semantic security property.

Average-Case Secure Obfuscation Our new definition only requires obfuscation for a *random* circuit in a family of circuits. Goldwasser and Kalai [12] considered this relaxed requirement and observed that it remains meaningful for the many cryptographic applications of obfuscation in which the circuit to be obfuscated *is* chosen at random. In fact, variants of this relaxation were considered (implicitly and explicitly) in the works of [7,8,10,14]. Normally the random choice of a circuit corresponds to the random selection of cryptographic keys. In this work we use the notion of *average-case secure obfuscation*: combining the more security-aware provisions suggested by [15,17] with the average-case relaxation (and considering probabilistic circuits).

1.2. *The Obfuscated Re-Encryption Scheme*

Comparison with Prior Work Mambo and Okamoto [16] noted the prevalence of re-encryption programs in practical applications and suggested efficiency improvements over the decrypt-and-encrypt approach. Blaze, Bleumer, and Strauss introduced the notion of *proxy re-encryption* [3,4] in which the re-encryption program is executed by a third-party *proxy*. In their security notion, the proxy cannot decrypt the messages of either Alice or Bob. The Blaze et al. construction is *bidirectional* (i.e., a program to translate ciphertexts from Alice to Bob can also be used to translate from Bob to Alice) and can be repeatedly applied (i.e., a ciphertext can be re-encrypted from Alice to Bob to Carol, etc.). Ateniese, Fu, Green, and Hohenberger [1] presented a semantic-security style definition for proxy re-encryption and designed the first *unidirectional* scheme, although it can only be applied once. Ateniese et al. also built a secure distributed storage system using their algorithms.

While these prior works are secure under specialized definitions, they cannot be considered as obfuscations for re-encryption since they leak subtle non-black-box information. On the other hand, the re-encryption definitions of Ateniese et al. [1] provide some security guarantee with respect to *dependent* auxiliary inputs, which we will not consider in this work. For example, they show that even when Alice has the re-encryption program from Alice to Bob, Bob's semantic security still holds. Our definition does not require this, but the scheme we present also has this property.

Overview of the Construction We now provide intuition behind our construction of an obfuscator for re-encryption (see Sect. 4 for the full construction). In a series of attempts, we develop a cryptosystem and an obfuscated re-encryption program which translates ciphertexts under pk_1 to ciphertexts under pk_2. Our starting point is a suitable public key cryptosystem.

Recall the semantically secure encryption scheme due to Boneh, Boyen, and Shacham [5] as instantiated in a group $\mathbb{G}$ of prime order q equipped with a bilinear map $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ where the Decision Linear problem is hard (see Sect. 3.1). The keys in this scheme are generated by selecting a random $h \xleftarrow{r} \mathbb{G}$ and $a, b \xleftarrow{r} \mathbb{Z}_q$, and setting $sk = (a, b, h)$ and $pk = (h^a, h^b, h)$. To encrypt a message $m \in \mathbb{G}$, select two random

values $r, s \xleftarrow{r} \mathbb{Z}_q$ and output the ciphertext $C = [h^{ar}, h^{bs}, h^{r+s} \cdot m]$. To decrypt a ciphertext $C = [W, X, Y]$, compute the plaintext $Y/(W^{1/a} \cdot X^{1/b})$. Let $pk_1 = (g^{a_1}, g^{b_1}, g)$ and $pk_2 = (h^{a_2}, h^{b_2}, h)$ be two public keys for this cryptosystem.

The basic (naive) re-encryption program from pk_1 to pk_2 contains (sk_1, pk_2). The program simply decrypts the input using sk_1 and encrypts the resulting message with pk_2. Clearly this program exposes both sk_1 and the underlying plaintext to any third party executing the re-encryption program.

As a first attempt to obfuscate the basic program, consider the re-encryption program that contains $Z_1 = a_2/a_1$ and $Z_2 = b_2/b_1$ and re-encrypts the ciphertext $[W, X, Y]$ by computing $[W^{Z_1}, X^{Z_2}, Y]$ for pk_2. (In a different context, a similar approach was suggested by Blaze et al. [4].) Unfortunately, this re-encryption program leaks non-black-box information (i.e., does not satisfy the virtual black-box property in Definition 2.2). For example, the program containing (Z_1, Z_2) which translates ciphertexts from Alice to Bob can be transformed into a new program containing (Z_1^{-1}, Z_2^{-1}) which translates ciphertexts from Bob to Alice—an operation which black-box access does not allow.

As a second attempt, consider the re-encryption program containing $Z_1 = h^{a_2/a_1}$ and $Z_2 = h^{b_2/b_1}$. Alice, with $sk_1 = (a_1, b_1, g)$, can compute this program given Bob's public key $pk_2 = (h^{a_2}, h^{b_2}, h)$. (On a different cryptosystem, a similar approach was suggested by Ateniese et al. [1].) The re-encryption program works as follows: on input a ciphertext $[W, X, Y] = [g^{a_1 r}, g^{b_1 s}, g^{r+s} \cdot m]$ under pk_1, output the ciphertext $[\mathbf{e}(W, Z_1), \mathbf{e}(X, Z_2), \mathbf{e}(Y, h)] = [E, F, G]$ under pk_2. To decrypt $[E, F, G]$, the holder of sk_2 would first compute $Q = G/(E^{1/a_2} \cdot F^{1/b_2})$ and then find and output the message m_i in the message space M such that $\mathbf{e}(m_i, h) = Q$. Of course, to ensure efficient decryption, this limits the size of the message space M to be polynomial. Notice the encryption scheme now supports two "forms" of ciphertexts—an original form and a re-encrypted one, each containing elements from different groups. As a result, a re-encrypted ciphertext cannot be further re-encrypted. The question, though, is whether or not such a program is any closer to being an obfuscation.

To meet the definition of secure obfuscation (Definition 2.2), the output of an adversary who is given the obfuscated program must be indistinguishable—even to a distinguisher with oracle access to the re-encryption program—from the output of a simulator given only black-box access to the program. Unfortunately, in the second attempt, knowledge of the public keys $pk_1 = (g^{a_1}, g^{b_1}, g)$ and $pk_2 = (h^{a_2}, h^{b_2}, h)$ easily allows a distinguisher to test whether a program containing (Z_1, Z_2) is a valid re-encryption program for these keys by checking that $\mathbf{e}(g^{a_1}, Z_1) = \mathbf{e}(g, h^{a_2})$ and $\mathbf{e}(g^{b_1}, Z_2) = \mathbf{e}(g, h^{b_2})$. We do not know how to construct a simulator that can output a program which passes this test.

To bypass this problem, we design our re-encryption program to be a probabilistic function of the keys. More specifically, consider the program containing $(y^{a_2/a_1}, y^{b_2/b_1}, y) = (Z_1, Z_2, Z_3)$ for a randomly selected $y \in \mathbb{G}$. (In the context of point function obfuscation, a similar approach was suggested by Canetti [7].) Alice can still generate this re-encryption program using only Bob's public key. The re-encryption program becomes: on input $[W, X, Y] = [g^{a_1 r}, g^{b_1 s}, g^{r+s} \cdot m]$ under pk_1, output the ciphertext under pk_2 as $[\mathbf{e}(W, Z_1), \mathbf{e}(X, Z_2), \mathbf{e}(Y, Z_3), Z_3] = [E, F, G, H]$. Decryption works as follows: first compute $Q = G/(E^{1/a_2} \cdot F^{1/b_2})$ and then output message m_i in the message space M such that $\mathbf{e}(m_i, H) = Q$.

This solution has one subtle problem because *all* ciphertexts produced by the obfuscated re-encryption program include $H = y$ as the fourth component, whereas ciphertexts produced by the decrypt-and-encrypt approach contain a fresh random value in that position. Thus, the obfuscated program does not "preserve the functionality" of the original one. This is easily fixed by having the obfuscated program re-randomize its output by choosing $z \xleftarrow{r} \mathbb{Z}_q$ and outputting $[E^z, F^z, G^z, H^z]$. (Note, it is not sufficient that we choose y randomly, since this choice is only made once for all re-encrypted ciphertexts, whereas z is chosen freshly for each re-encryption.)

Even this, however, falls short, because we do not know how to prove this construction is secure. In particular, since the distinguisher has access to a re-encryption oracle, it can query the oracle on the values contained in the obfuscated program. Indeed, in the above scheme, there is a specific property of valid obfuscated programs that a distinguisher can test for, and we do not know how to construct a simulator that also passes this test. Precisely, this test is as follows. On input a program (Z_1, Z_2, Z_3), the distinguisher queries his re-encryption oracle on the "ciphertext" $[Z_3, Z_2, Z_1]$ and obtains the output $[E, F, G, H]$. Then, if and only if $E = G$, the distinguisher guesses that (Z_1, Z_2, Z_3) is a valid obfuscated program. (A valid obfuscation will always pass this test; the re-encryption program output by our simulator is comprised of three group elements chosen uniformly at random and thus is unlikely to pass this test.)

In order to overcome this final hurdle, our next and final re-encryption program re-randomizes the input ciphertext before applying the transformation above. If the public key is (g^a, g^b, g) and the input ciphertext is $C = [W, X, Y]$, our re-encryption program re-randomizes C by sampling r', s' and computing the ciphertext $[W \cdot (g^a)^{r'}, X \cdot (g^b)^{s'}, Y \cdot g^{r'+s'}]$. Finally, we are able to show this construction meets our obfuscation definition under two reasonable complexity assumptions.

As a final point about our complexity assumptions, because our obfuscation definition only requires average-case obfuscation, we do not have to make the strong complexity assumptions necessary in the constructions of Canetti [7] and Wee [20], which can be broken in quasi-polynomial time. Thus, our scheme simultaneously meets a strong theoretical definition while retaining the sensibility associated with standard assumptions.

Organization In Sect. 2, we present the definition of obfuscation and explain in more detail how it captures both obfuscation and security for many cryptographic functionalities. In Sect. 3, we introduce some notation and complexity assumptions. In Sect. 4, we present our encryption scheme and a family of re-encryption circuits and finally, in Sect. 5, we present our obfuscator for re-encryption circuits.

2. Definitions

Barak et al. [2] required that an obfuscator strip programs of non-black-box information. They formalized this by requiring that any predicate computable from the obfuscated program is also computable from black-box access to it. Goldwasser and Kalai [12] gave a stronger definition, guaranteeing security in the presence of (dependent and independent) auxiliary input. A formal definition, which we call *predicate black-box obfuscation* (or predicate obfuscation, for short), follows.

By $\mu(k)$ we denote some *negligible* function, i.e., one such that, for all $c > 0$ and all sufficiently large k, $\mu(k) < 1/k^c$. For a family $\mathbf{C}$ of polynomial-size circuits and for a length parameter k, let $\mathbf{C}_k$ be the circuits in $\mathbf{C}$ with input length k; we write $\mathbf{C} = \{\mathbf{C}_k\}$ as shorthand for such a partition. For an Oracle Turing Machine M and a circuit C, let $M^C(x)$ denote the machine M activated on an input x with an oracle to the function computed by C.

Definition 2.1 (Predicate Obfuscation [2,12]). An efficient algorithm Obf is a *predicate obfuscator for the family* $\mathbf{C} = \{\mathbf{C}_k\}$, if it satisfies the following properties:

- Preserving Functionality: There exists a negligible function $\mu(k)$, s.t. for all input lengths k, for any $C \in \mathbf{C}_k$:

$$\Pr\left[\exists x \in \{0,1\}^k : \big(\mathsf{Obf}(C)\big)(x) \neq C(x)\right] \leq \mu(k).$$

  The probability is taken over the random coins used by Obf.
- Polynomial Slowdown: There exists a polynomial $p(k)$ such that for sufficiently large input lengths n, for any $C \in \mathbf{C}_k$, the obfuscator Obf only enlarges C by a factor of p: $|\mathsf{Obf}(C)| \leq p(|C|)$.
- Predicate Virtual Black-box: For every polynomial-sized adversary circuit $\mathcal{A}$, there exists a polynomial-size simulator circuit Sim and a negligible function $\mu(k)$, such that for every input length n, for every $C \in \mathbf{C}_k$, for every predicate π, for every auxiliary input $z \in \{0,1\}^{q(k)}$:

$$\left|\Pr\left[\mathcal{A}\big(\mathsf{Obf}(C), z\big) = \pi(C, z)\right] - \Pr\left[\mathsf{Sim}^C\big(1^k, z\big) = \pi(C, z)\right]\right| \leq \mu(k).$$

  The probability is over the coins of the adversary, the simulator and the obfuscator.

As discussed in Sect. 1, the predicate black-box definition does not guarantee *security* when obfuscated circuits are used in cryptographic settings. Following a definition suggested by Pass [17] (and similarly to a definition suggested independently by Hofheinz et al. [15]), we use a *composable* notion of obfuscation. Moreover, we particularly want to focus on the *average-case* security of obfuscating probabilistic circuits.

Definition 2.2 (Average-Case Secure Obfuscation). An efficient algorithm Obf that runs on input a (probabilistic) circuit and outputs a new (probabilistic) circuit, is an *average-case secure obfuscator* for the family $\mathbf{C} = \{\mathbf{C}_k\}$, if it satisfies the following properties:

- Preserving Functionality: "With overwhelming probability $\mathsf{Obf}(C)$ behaves *almost* identically to C on all inputs". There exists a negligible function $\mu(k)$, such that for any input length k, for any $C \in \mathbf{C}_k$:

$$\Pr_{\text{coins of } \mathsf{Obf}}\left[\forall x \in \{0,1\}^k : \Delta\Big(\big(\mathsf{Obf}(C)\big)(x), C(x)\Big) \leq \mu(k)\right] \geq 1 - \mu(k).$$

  The distributions $(\mathsf{Obf}(C))(x)$ and $C(x)$ are taken over $\mathsf{Obf}(C)$'s and C's random coins respectively. Δ denotes the statistical distance between distributions.[3]

[3] Our construction of a re-encryption obfuscator *perfectly* preserves functionality, i.e. the output distributions of the original and obfuscated programs are identical.

- Polynomial Slowdown: (identical to Definition 2.1)
- Average-Case Secure Virtual Black-Box: For any efficient adversary $\mathcal{A}$, there exists an efficient simulator Sim and a negligible function $\mu(k)$, such that for every efficient distinguisher D, for every input length k and for every polynomial-size auxiliary input z:

$$\left|\begin{array}{l}\Pr\left[C \leftarrow \mathbf{C}_k : \mathsf{D}^C\left(\mathcal{A}\left(\mathsf{Obf}(C), z\right), z\right) = 1\right] \\ \quad - \Pr\left[C \leftarrow \mathbf{C}_k : \mathsf{D}^C\left(\mathsf{Sim}^C\left(1^k, z\right), z\right) = 1\right]\end{array}\right| \leq \mu(k).$$

  The probability is over the selection of a *random circuit* C from $\mathbf{C}_k$, and the coins of the distinguisher, the simulator, the oracle and the obfuscator. Note that entities with black-box access to C cannot set C's random tape.

For the last requirement, it is sufficient to require the existence of a simulator for the *simple* adversary that just outputs its own input. This would give an equivalent definition since any adversary's code can be incorporated into the distinguisher; indeed, Hofheinz et al. [15] take this approach (but present a slightly relaxed definition).

Discussion Intuitively, Definition 2.2 guarantees that any attack that an adversary can mount using the obfuscated circuit can also be mounted by a simulator with oracle access to the functionality. This definition differs from the predicate definition in several ways. It considers obfuscation of a *random* circuit from a family, and furthermore, the circuit families considered can be *probabilistic* (this allows us to side-step impossibility results, see Sect. 2.2). We also follow [12] in requiring that the obfuscation be secure in the presence of (independent) auxiliary input, where the auxiliary input is selected first, and then a random circuit is chosen from the family. Note that the average-case secure virtual black-box requirement of the above definition is incomparable to the predicate black-box requirement of [2,12]; the latter is weaker in that it only requires that the obfuscator hides *predicates*, but is stronger in that it provides the predicate distinguisher with the actual program (whereas our definition only gives our predicate distinguisher black-box access).

Finally, we emphasize that in this new definition there are *two* important sources of randomness. The first source of randomness stems from the fact that the circuits subject to obfuscation are probabilistic. The second, more subtle, source of randomness is in the selection of a random circuit C from the family $\mathbf{C}_k$. The average-case secure virtual black-box requirement guarantees security when a circuit is selected from the family by a *specific* distribution (i.e., the uniform distribution—one should think of this as uniformly choosing random keys for a cryptographic scheme). The predicate black-box definition, on the other hand, guarantees security for *every* circuit in the family, or (equivalently) for *every distribution* on circuits. Other work [8,10] guarantees security for a large class of distributions on circuits from a family, such as *every* distribution with at least super-logarithmic min-entropy. The above notion of secure obfuscation can be generalized to give security against more general classes of distributions. For clarity, we choose to present the less general definition above.

Comparison with Hofheinz, Malone-Lee and Stam Hofheinz, Malone-Lee and Stam [15] also seek a notion of obfuscation suitable for cryptographic applications; they independently suggested a similar definition of obfuscation. Their definition, however, is slightly more relaxed in its virtual black-box requirement. In our terms, they require that for every distinguisher D with oracle access, for every adversary $\mathcal{A}$ with obfuscated program access, there exists a simulator Sim with oracle access such that D cannot distinguish between the outputs of the adversary $\mathcal{A}$ and the simulator Sim. The difference between their definition and Definition 2.2 is that they allow the simulator to depend on the distinguisher, whereas Definition 2.2 requires the existence of a "Universal Simulator" that works for *all* distinguishers.

This relaxed definition is sufficient for the applications considered by Hofheinz et al. Naturally, their negative results apply also to obfuscation under Definition 2.2. In fact, their relaxed definition also suffices for proving the security of semantically secure re-encryption schemes. Nonetheless, our work focuses on the construction of an explicit obfuscator for a new functionality, and thus we work with the strongest definition that we can prove our construction meets.

2.1. *Meaningfulness for Security*

This section serves as a discussion of the security guarantee provided by average-case secure obfuscation. As mentioned in Sect. 1, the definition of obfuscation should be composable in the following sense: *"If a cryptographic scheme is secure when the adversary is given black-box access to a program, then it remains secure when the adversary is given the obfuscated program."* We claim that for several applications (including indistinguishable-secure re-encryption), average-case secure obfuscation indeed gives this guarantee.

Let $\mathbf{C} = \{\mathbf{C}_n\}$ be a family of circuits and let $\mathsf{Expt}^C(A^C, 1^k)$ be a probabilistic binary experiment relative to the circuit family $\mathbf{C}$ that involves an adversary A and a security parameter 1^k. In other words, the experiment is an algorithm that, when run on 1^k, given access to a randomly chosen oracle $C \in \mathbf{C}_k$, and given the representation of an adversary A that also makes calls to oracle C, eventually outputs either a zero or one. For example, indistinguishable-secure encryption—and even the enhanced notion of such encryption with access to a re-encryption oracle defined in Definition 4.3—is such an experiment relative to an oracle that outputs the public key of the encryption scheme. For simplicity of notation, we have not explicitly written the auxiliary input z that is also given to Expt and the adversary $\mathcal{A}$.

Theorem 2.3. *Let* $\mathbf{C} = \{\mathbf{C}_k\}$ *be a family of circuits. Let* $\mathsf{Expt}_0^C(A^C, 1^k)$ *and* $\mathsf{Expt}_1^C(A^C, 1^k)$ *be two probabilistic experiments with respect to oracle* $C \in \mathbf{C}_k$*, and let* Obf *be a average-case secure obfuscator for the a family of circuits* $\mathbf{C}$*. Suppose that for all p.p.t. adversaries* A*, it holds that*

$$\left\{C \leftarrow \mathbf{C}_k;\ \mathsf{Expt}_0^C\left(A^C, 1^k\right)\right\}_k \approx_s \left\{C \leftarrow \mathbf{C}_k; \mathsf{Expt}_1^C\left(A^C, 1^k\right)\right\}_k.$$

Then for all p.p.t. adversaries A' *that are also given an obfuscated version of circuit* C (*and therefore implicitly have oracle access to* C *as well*)*, it holds that*

$$\left\{C \leftarrow \mathbf{C}_k;\ \mathsf{Expt}_0^C\left(A'\left(\mathsf{Obf}(C)\right), 1^k\right)\right\}_k \approx_s \left\{C \leftarrow \mathbf{C}_k; \mathsf{Expt}_1^C\left(A'\left(\mathsf{Obf}(C)\right), 1^k\right)\right\}_k.$$

Proof. By the average-case secure obfuscation property, it holds that for all p.p.t. adversaries A',

$$\begin{aligned}&\Pr\left[C \leftarrow \mathbf{C}_k;\ \mathsf{Expt}_0^C\left(A'\left(\mathsf{Obf}(C)\right), 1^k\right) = 1\right] \\ &\quad = \Pr\left[C \leftarrow \mathbf{C}_k;\ \mathsf{Expt}_0^C\left(A'\left(\mathsf{Sim}^C\right), 1^k\right) = 1\right] \pm \mu(k)\end{aligned}$$

for some negligible function μ. This follows syntactically because Expt_0 can be viewed as a distinguisher algorithm. Let A^C be the adversary that first runs Sim^C, and then runs A' on the resulting output. Algorithm A can use its own oracle C to answer any oracle queries that A' makes. Thus, we have by definition that

$$\Pr\left[C \leftarrow \mathbf{C}_k;\ \mathsf{Expt}_0^C\left(A'\left(\mathsf{Sim}^C\right), 1^k\right) = 1\right] = \Pr\left[C \leftarrow \mathbf{C}_k;\ \mathsf{Expt}_0^C\left(A^C, 1^k\right) = 1\right].$$

Using the hypothesis on Expt_0 and Expt_1 for any p.p.t. A, it follows that

$$\begin{aligned}&\Pr\left[C \leftarrow \mathbf{C}_k;\ \mathsf{Expt}_0^C\left(A^C, 1^k\right) = 1\right] \pm \mu(k) \\ &\quad = \Pr\left[C \leftarrow \mathbf{C}_k;\ \mathsf{Expt}_1^C\left(A^C, 1^k\right) = 1\right] \pm \mu'(k) \\ &\quad = \Pr\left[C \leftarrow \mathbf{C}_k;\ \mathsf{Expt}_1^C\left(A'\left(\mathsf{Obf}(C)\right), 1^k\right) = 1\right] \pm \mu'(k)\end{aligned}$$

for some negligible function μ'. The second line follows by the definition of A and establishes the theorem. □

Discussion of Theorem 2.3 In order to apply the notion of average-case obfuscation (in the first line of the proof), it is important that the oracle given to the experiment is the same as the obfuscated circuit given to the adversary. For example, this explains the difficulty in constructing a CCA2-secure public key encryption scheme by obfuscating a CCA2-secure private encryption scheme. The CCA2 experiment (and in particular, the adversary) requires access to a decryption oracle. However, the obfuscated CCA2-private encryption circuit would only provide encryption functionality. Because the experiment circuit and the obfuscated one mismatch, this proof would not apply.[4] This was also observed by Hofheinz et al. [15].

Nonetheless, for primitives with security experiments that can be simulated with access to a circuit oracle, Definition 2.2 does guarantee that for every adversary that mounts an attack using an obfuscated circuit, there exists a *black-box* simulator that can mount an attack with a similar success probability. Thus, if the scheme is secure against an adversary with black-box access to a circuit C, it is also secure against an adversary with an obfuscated version of C. It is important to note that the predicate definition of obfuscation (i.e., Definition 2.1) would *not* let us make the above conclusion.

[4] It is, however, possible to formulate a more delicate definition that considers giving the distinguisher access to different oracles, such as a decryption oracle for CCA-security or a signing oracle for signatures. In this work, we focus on oracles which we know how to implement via an obfuscated circuit. A public key oracle is trivial to implement; the value can simply be encoded into the wiring of the circuit. We provide a re-encryption obfuscator in this work. Unfortunately, it seems implausible that such obfuscators exist for CCA-secure decryption or digital signatures [12].

2.2. *Obfuscating Probabilistic Programs*

In this section we discuss an impossibility result for average-case secure obfuscation of *deterministic* circuits, and explain how we side-step this impossibility by considering probabilistic circuits. Wee [20] observes that the only deterministic circuits that can be obfuscated under strong composable notions of obfuscation are those that are (exactly) *learnable*. A similar result also applies to obfuscating deterministic circuits under Definition 2.2. To see this, we first define what it means for a family of circuits to be *learnable*:

Definition 2.4 (Approximate Learnability, see Valiant [19]). A family of *deterministic* circuits $\mathbf{C} = \{\mathbf{C}_k\}$ is approximately learnable (on average) if there exists a learning algorithm Ł that runs on input a polynomial function $\varepsilon(k)$ and black-box access to a random $C \in \mathbf{C}_k$, and with all but negligible probability (over the selection of C and the coins of L), outputs a circuit C', such that $\Pr_r[C(r) \neq C'(r)] < \varepsilon(k)$ where r is a uniformly random input in $\{0,1\}^k$. Furthermore, if L's running time is polynomial in $(n, \frac{1}{\varepsilon(k)})$ then we say that $\mathbf{C}$ is *efficiently* approximately learnable.

Now, following the intuition of Wee [20], we observe that any circuit family that is obfuscatable under Definition 2.2 is also efficiently approximately learnable. This was also observed independently by Hofheinz, Malone-Lee and Stam [15] and Pass [17].

Proposition 2.5. *If a family of* deterministic *circuits* $\mathbf{C} = \{\mathbf{C}_k\}$ *is obfuscatable under Definition* 2.2 *(even inefficiently),*[5] *then it is also efficiently approximately learnable.*

Proof. Consider the "dummy" adversary Adv that simply outputs the obfuscated circuit $\mathsf{Obf}(C)$ it receives as input, and a distinguisher D (with black-box access to $C \in \mathbf{C}_k$) that on input a circuit C' and auxiliary information z, generates $O(\frac{k}{\varepsilon(k)})$ uniformly random inputs, and outputs 1 if C' and C agree on these inputs, and 0 otherwise.[6]

Because Obf preserves functionality, the above adversary Adv causes the above distinguisher D to output 1 with high probability (over the coins of Obf and D). To make the distinguisher accept with similar probability, the simulator S for the empty adversary must learn, from black-box access to C, a circuit that works (at the very least) very similarly to C on random inputs. More formally, with all but negligible probability (over the selection of $C \in \mathbf{C}_n$ and the simulator's coins), the output distribution of S with black-box access to C and the output distribution of C (on a random input) should have statistical distance at most $\varepsilon(n)$. Otherwise, D distinguishes between the outputs of the adversary and the simulator with probability close to 1. Thus S is an efficient algorithm for approximately learning the circuit family $\mathbf{C}$. Finally, note that the simulator S is an efficient learning algorithm regardless of whether or not the obfuscator Obf is efficient. □

The above impossibility disappears when we consider probabilistic circuit families. This is because the (efficient) distinguisher with black-box access to a probabilistic C

[5] An "inefficient" obfuscator may run in time that is super-polynomial in the size of the circuit being obfuscated.

[6] Wee [20] considers different distinguishers that check different inputs.

and non-black-box access to $\mathsf{Obf}(C)$ cannot necessarily distinguish whether the *distributions* that C and $\mathsf{Obf}(C)$ output on a particular input are *statistically* close or far. This is similar to the case of encryption (see Goldwasser and Micali [13]), where only randomness can prevent an adversary from recognizing whether two ciphertexts are encryptions of the same bit. Our obfuscation of re-encryption programs uses this observation. In fact, the re-encryption simulator we construct (for the empty adversary) outputs a "dummy circuit" that has little to do with the circuit being obfuscated, but is still indistinguishable from the true obfuscated circuit.

3. Algebraic Setting and Assumptions

Definition 3.1 (Computational and Statistical Indistinguishability). Two ensembles of probability distributions $\{X_k\}_{k\in\mathbb{N}}$ and $\{Y_k\}_{k\in\mathbb{N}}$ with index set $\mathbb{N}$ are said to be *computationally indistinguishable* if for every polynomial-size circuit family $\{D_k\}_{k\in\mathbb{N}}$, there exists a negligible function μ such that

$$\left|\Pr\left[x \leftarrow X_k : D_k(x) = 1\right] - \Pr\left[y \leftarrow Y_k : D_k(y) = 1\right]\right| < \mu(k).$$

We denote such sets $\{X_k\}_{k\in\mathbb{N}} \overset{c}{\approx} \{Y_k\}_{k\in\mathbb{N}}$. $\{X_k\}_{k\in\mathbb{N}}$ and $\{Y_k\}_{k\in\mathbb{N}}$ are *statistically indistinguishable* if the above holds for every (not necessarily polynomial-size) circuit family $\{D_k\}_{k\in\mathbb{N}}$. We denote such sets $\{X_k\}_{k\in\mathbb{N}} \overset{s}{\approx} \{Y_k\}_{k\in\mathbb{N}}$.

A fact which we will later use is that computational indistinguishability for a single bit implies statistical indistinguishability.

3.1. *Bilinear Maps*

Let $\mathsf{BMsetup}$ be an algorithm that, on input the security parameter 1^k, outputs the parameters for a bilinear map as $(q, g, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$, where $\mathbb{G}, \mathbb{G}_T$ are groups of prime order $q \in \Theta(2^k)$ and g is a generator of $\mathbb{G}$. The efficient mapping $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is both *bilinear*, i.e., for all $h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_q$, $\mathbf{e}(h^a, h^b) = \mathbf{e}(h, h)^{ab}$, and *non-degenerate*, i.e., if g generates $\mathbb{G}$, then $\mathbf{e}(g, g) \neq 1$. In such a group, observe that for all $g, h \in \mathbb{G}$, $e(g^a, h^b) = e(g, h)^{ab}$ and $e(g, h)^a = e(g^a, h) = e(g, h^a)$.

For simplicity, we present our solution using bilinear maps of the form $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Our scheme can also be implemented in the more general setting where $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and isomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$ may not be efficiently computable. In either case, there are known efficient bilinear and non-degenerate mappings; see Galbraith, Paterson, and Smart [11] for more information on various implementation options.

Complexity Assumptions We make the following complexity assumptions in bilinear groups.

Assumption 3.2 (Strong Diffie Hellman Indistinguishability). Let $\mathbb{G}$ be a group of order q where q is a k-bit prime, $g \overset{r}{\leftarrow} \mathbb{G}$ and $a, b, c, d \overset{r}{\leftarrow} \mathbb{Z}_q$. Then the following two

distributions are computationally indistinguishable:

$$\left\{g, g^a, g^b, g^c, g^{abc}\right\}_k \overset{c}{\approx} \left\{g, g^a, g^b, g^c, g^d\right\}_k.$$

Boneh, Sahai and Waters [6] previously proposed a similar assumption, called Decision 3-party Diffie–Hellman, where the above must hold in a prime subgroup of a composite order bilinear group.

Assumption 3.3 (Decision Linear [5]). Let $\mathbb{G}$ be a group of order q where q is a k-bit prime, $f, g, h \overset{r}{\leftarrow} \mathbb{G}$ and $a, b, c \overset{r}{\leftarrow} \mathbb{Z}_q$. Then the following two distributions are computationally indistinguishable:

$$\left\{f, g, h, f^a, g^b, h^{a+b}\right\}_k \overset{c}{\approx} \left\{f, g, h, f^a, g^b, h^c\right\}_k.$$

4. A Special Encryption Scheme and Re-Encryption Functionality

4.1. *A Special Encryption Scheme* Π

In this section, we first recall the definition of an encryption scheme, and then describe a special semantically secure encryption scheme and a *re-encryption functionality* for which we later present a secure obfuscation scheme. For convenience of our notation, the encryption algorithm supports two forms of ciphertexts and takes an additional input $\beta \in \{0, 1\}$ to choose between them.

Definition 4.1 (Two-Format Encryption Algorithms). An encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with two different encryption formats for message space $M = \{0, 1\}^{p(k)}$, where p is a polynomial and 1^k is the security parameter, is comprised of the following algorithms:

- $\mathsf{KeyGen}(1^k)$: the key generation algorithm takes in the security parameter 1^k and outputs a key pair (pk, sk).
- $\mathsf{Enc}(pk, \beta \in \{0, 1\}, m \in M)$: the encryption algorithm takes in a public key pk, a format β, and a message m, and produces a ciphertext c.
- $\mathsf{Dec}(sk, c)$: the decryption algorithm takes in a secret key sk and a ciphertext c, and outputs either a message $m \in M$ or the error symbol $\perp$.

Our special encryption scheme Π is described in Fig. 1. For the first form, encryption and decryption work as per the Boneh, Boyen, and Shacham [5] construction. For the second form, the encryption and decryption are novel and relevant for re-encryption. Note that this encryption system also requires the message space M to be a subset of $\mathbb{G}$ whose size is a polynomial in k. As we argue in Lemma 4.4, larger messages can be handled by standard block-by-block composition. Theorem 4.5 appearing below establishes the semantic security of scheme Π.

Lemma 4.2. *The Encryption scheme* Π *is perfectly complete.*

Common: For a security parameter 1^k, let $(q, g, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathsf{BMsetup}(1^k)$ be a common parameter and let $M \subset \mathbb{G}$ where $|M| = O(\text{poly}(k))$ be the message space.

$\mathsf{KeyGen}(1^k, (q, g, \mathbb{G}, \mathbb{G}_T, \mathbf{e}))$:

1. Randomly select a new generator $h \xleftarrow{r} \mathbb{G}$ and random $a, b \xleftarrow{r} \mathbb{Z}_q$.
2. Output $pk = (h^a, h^b, h)$ and $sk = (a, b, h)$.

$\mathsf{Enc}(pk, \beta, m)$:

1. Parse $pk = (h^a, h^b, h)$.
2. Choose random $r, s \xleftarrow{r} \mathbb{Z}_q$.
3. If $\beta = 0$, output the ciphertext $[0,\ (h^a)^r,\ (h^b)^s,\ h^{r+s} \cdot m,\ 0] \in \mathbb{Z} \times \mathbb{G}^3 \times \mathbb{Z}$.
4. If $\beta = 1$, first choose a random element $t \xleftarrow{r} \mathbb{G}$, and then output the ciphertext $[1,\ \mathbf{e}((h^a)^r,\ t),\ \mathbf{e}((h^b)^s,\ t),\ \mathbf{e}(h^{r+s} \cdot m,\ t),\ t] \in \mathbb{Z} \times \mathbb{G}_T^3 \times \mathbb{G}$.

$\mathsf{Dec}(sk, [s, W, X, Y, Z])$:

1. Parse $sk = (a, b, h)$.
2. If $s = 0$, then output $Y/(W^{1/a} \cdot X^{1/b})$.
3. If $s = 1$, then
   (a) Compute $Q = Y/(W^{1/a} \cdot X^{1/b})$.
   (b) For each $m \in M$, test if $\mathbf{e}(m, Z) = Q$. If so, output m and halt.

Fig. 1. Encryption scheme Π.

Proof. When $\beta = 0$, this follows from the completeness property of the Boneh, Boyen, and Shacham [5] scheme. When $\beta = 1$, on input $[1, E, F, G, H]$ (produced by the honest encryption algorithm), the decryption algorithm first computes $Q = \frac{G}{E^{1/a} \cdot F^{1/b}}$. Notice that

$$\begin{aligned} Q &= \frac{\mathbf{e}(h^{r+s} \cdot m,\ H)}{\mathbf{e}((h^a)^r,\ H)^{1/a} \cdot \mathbf{e}((h^b)^s,\ H)^{1/b}} \\ &= \frac{\mathbf{e}(h^{r+s} \cdot m,\ H)}{\mathbf{e}(h^r,\ H) \cdot \mathbf{e}(h^s,\ H)} \\ &= \mathbf{e}(m, H). \end{aligned}$$

The decryption algorithm loops over each (of the polynomially many) $m_i \in M$ and tests whether $\mathbf{e}(m_i, H) = Q$ and therefore always recovers m as required. □

4.2. *Re-Encryption Functionality*

Recall that obfuscation is with respect to a class of circuits. We now define a special class of re-encryption circuits for the encryption scheme Π which can be easily analyzed.

Let (pk_1, sk_1) and (pk_2, sk_2) be two keys pairs generated by running KeyGen on independent random tapes. When given a ciphertext generated by algorithm Enc using pk_1,

a *re-encryption circuit* decrypts the ciphertext and then re-encrypts the resulting message under a second public key pk_2. For technical reasons, we also require the circuit to produce the pairs of public keys for which it transforms ciphertexts. More formally, the re-encryption functionality is as follows:

$$\text{On input } x, \text{ output } \begin{cases} (pk_1, pk_2) & \text{if } x = \texttt{keys}; \\ \perp & \text{if } \mathsf{Dec}(sk_1, x) = \perp; \\ \mathsf{Enc}(pk_2, 1, \mathsf{Dec}(sk_1, x)) & \text{otherwise.} \end{cases}$$

That is, on special input `keys`, it outputs the ordered pair of public keys (pk_1, pk_2). Any other input is considered to be a (first-form) ciphertext; it is decrypted using sk_1 to produce either a message m or an indication that the input is ill-formed. In the latter case, the output is $\perp$. When the decryption is successful (resulting in a message m), the output is the (second-form) ciphertext $c \leftarrow \mathsf{Enc}(pk_2, 1, m)$. Furthermore, we require that the circuit encoding this functionality, C_{sk_1,pk_2}, allows the values pk_1 and pk_2 to be read from the circuit description. This property is easy to achieve by adding a "data" section to the circuit which does not affect the circuit's output, but encodes a message, with say, AND gates encoding a 1 and OR gates encoding a 0. We now define the family of circuits:

$$C_k = \left\{ C_{sk_1,pk_2} \mid (pk_1, sk_1) \leftarrow \mathsf{KeyGen}\left(1^k\right), (pk_2, sk_2) \leftarrow \mathsf{KeyGen}\left(1^k\right) \right\}.$$

4.3. *Security for Re-Encryption*

Recall from Sect. 2.1 that the goal of this work is to formalize and illustrate *composable* obfuscation, where any system secure with respect to a black-box program retains its security when the black-box access is replaced by access to an obfuscated program. We now want to argue that the encryption scheme in Fig. 1 is secure with respect to a black-box re-encryption program. We will later show a composable obfuscation for this re-encryption program. These two results will allow us to conclude that the security of the encryption scheme is not compromised by the release of an obfuscated program.

We generalize the standard notion of indistinguishability [13] for encryption schemes by allowing the adversary to have access to a re-encryption oracle. In particular, the following definition captures the notion that "given a ciphertext y and black-box access to a re-encryption circuit, an adversary does not learn any information about the plaintext corresponding to y."

Definition 4.3 (IND-security with Oracle C_{sk_1,pk_2}). Let $\Gamma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a two-format encryption scheme for the message space M and let the random variable $\mathsf{IND}_b(\Gamma, \mathcal{A}, k)$ where $b \in \{0, 1\}$, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and $k \in \mathbb{N}$ denote the result of the following probabilistic experiment:

$\mathsf{IND}_b(\Gamma, \mathcal{A}, k)$
$(pk_1, sk_1) \leftarrow \mathsf{KeyGen}\left(1^k\right)$, $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}\left(1^k\right)$,
$(m_0, m_1, i, \beta, z) \leftarrow \mathcal{A}_1^{C_{sk_1,pk_2}}\left(1^k\right)$ s.t. $m_0, m_1 \in M$,
$y \leftarrow \mathsf{Enc}(pk_i, \beta, m_b)$,
$B \leftarrow \mathcal{A}_2^{C_{sk_1,pk_2}}(y, z)$,
Output B.

Encryption scheme Γ is indistinguishable secure with oracle C_{sk_1,pk_2} if $\forall$ p.p.t. algorithms $\mathcal{A}$ the following two ensembles are computationally indistinguishable:

$$\left\{\mathsf{IND}_0(\Gamma,\mathcal{A},k)\right\}_k \stackrel{c}{\approx} \left\{\mathsf{IND}_1(\Gamma,\mathcal{A},k)\right\}_k.$$

When the added specificity of the oracle C_{sk_1,pk_2} is not needed, we may say that Γ is indistinguishable secure with a re-encryption oracle.

For simplicity, we allow the adversary to pick the key pk_i under which the challenge is encrypted and the form β of the encryption. By a standard hybrid argument, the above definition is equivalent to one in which the adversary is given four encryptions of the challenge message—one per key and form.

Larger Messages Our construction uses a messages spaces M such that $|M| = \text{poly}(k)$. The following lemma, however, shows that this property is not troublesome, because a standard hybrid argument can be used to show that even single-bit schemes can be used to produce schemes that handle larger message spaces.

Lemma 4.4. *If there exists an encryption scheme $\Gamma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ that is IND-secure with a re-encryption oracle for security parameter 1^k and message space $M = \{0,1\}^c$ for some constant c, then there exists a scheme Γ' that is IND-secure with a re-encryption oracle for message space $M = \{0,1\}^{c\cdot p(k)}$ for any polynomial p.*

Proof. The proof follows from a standard hybrid argument, and so it is only sketched here. Let Γ' be the same as Γ except that Enc' calls the Enc routine $p(k)$ times to encrypt each group of c bits of the message independently, and Dec' similarly decrypts the ciphertext in a block-by-block fashion. The re-encryption oracle, $C_{pk'_1,sk'_2}$ also handles $c \cdot p(k)$-bit messages by working in a block-by-block manner.

Suppose Γ' was not indistinguishable secure with a re-encryption oracle. In other words, there exists an efficient adversary algorithm A and distinguisher algorithm D such that for infinitely many k, it holds that $|\Pr[D(\mathsf{IND}_0(\Gamma',\mathcal{A},k)) = 1] - \Pr[D(\mathsf{IND}_1(\Gamma',\mathcal{A},k) = 1)]| > \frac{1}{k^d}$ for some constant d. Let us define an experiment $\mathsf{IND}_{0,i}(\Gamma',\mathcal{A},k)$ which is the same as IND_0 with the exception that when the challenge ciphertext is computed, it is not computed on m_0, but rather on the ciphertext $m_{0,i}$ in which the first $i \cdot c$ bits are from m_0 and the remaining bits of the message come from m_1. Notice that $\mathsf{IND}_{0,p(k)} = \mathsf{IND}_0$ and $\mathsf{IND}_{0,0} = \mathsf{IND}_1$. Thus there must exist some i such that $|\Pr[D(\mathsf{IND}_{0,i}(\Gamma',\mathcal{A},k)) = 1] - \Pr[D(\mathsf{IND}_{0,i+1}\Gamma',\mathcal{A},k) = 1]| > \frac{1}{k^d\,p(k)}$. From this statement, it follows how $\mathcal{A}$ can be used to break the security of the c-bit scheme Γ. □

Theorem 4.5. *The encryption scheme Π (in Fig. 1) is an indistinguishable-secure encryption scheme with oracle C_{sk_1,pk_2} under the Decision Linear assumption in $\mathbb{G}$.*

Proof. To show that scheme Π meets security Definition 4.3, suppose adversary $\mathcal{A} = (\mathcal{A}_1,\mathcal{A}_2)$ and distinguisher D has advantage ε in distinguishing $\mathsf{IND}_0(\Pi,\mathcal{A},k)$ from $\mathsf{IND}_1(\Pi,\mathcal{A},k)$. Then, we construct an adversary $\mathcal{A}'$ that decides the decision linear problem with advantage $\varepsilon/2$ as follows. Let $\Gamma = (h_1,h_2,h,h_1^x,h_2^y,Q)$ be an instance of the decision linear problem; $\mathcal{A}'$ on input Γ works as follows:

1. Sample $a, b, c \xleftarrow{r} \mathbb{Z}_q$.
2. Set $pk_1 = (h_1, h_2, h)$ and $pk_2 = (h_1^{ac}, h_2^{bc}, h^c)$.
3. Sample $v \in \mathbb{G}$ and compute the values $Z_1 \leftarrow v^a$, $Z_2 \leftarrow v^b$ and $Z_3 \leftarrow v$. Generate circuit $R_{pk_1,pk_2,Z}$ as per the description in Fig. 2 below.
4. Run $\mathcal{A}_1^{\mathcal{O}}(1^k)$ to produce a tuple (m_0, m_1, i, β, z).
   When $\mathcal{A}$ queries its oracle on the tuple $[s, W, X, Y, Z]$, respond as follows:
   (a) Return $\perp$ if $s \neq 0$ or $Z \neq 0$, or if $W, X, Y \notin \mathbb{G}$, etc.
   (b) Otherwise, run $R_{pk_1,pk_2,Z}$ on the tuple and return the result.
5. Sample a bit $t \xleftarrow{r} \{0, 1\}$.
6. Set y to be the ciphertext $[0, h_1^x, h_2^y, Q \cdot m_t, 0]$ if $i = 0$ and $[0, (h_1^x)^{ac}, (h_2^y)^{bc}, Q^c \cdot m_t, 0]$ if $i = 1$. Furthermore, if $\beta = 1$, and $y = [0, W, X, Y, 0]$, then choose a random $s \leftarrow \mathbb{G}$ and recompute y as the tuple $[1, \mathbf{e}(W, s), \mathbf{e}(X, s), \mathbf{e}(Y, s), s]$.
7. Run $B \leftarrow \mathcal{A}_2^{\mathcal{O}}(y, z)$ and respond to the oracle queries as above.
8. Run $t' \leftarrow D(B)$ and output 1 if $t' = t$ (i.e., guess that Γ is a DLA instance) and otherwise output 0.

We argue that when Γ is a proper decision linear instance, $\mathcal{A}'$ perfectly simulates the experiment IND_t. First, the distribution of keys created in step 1 and 2 is identical to distribution of keys in the first line of IND_t. The remaining lines of IND_t are also syntactically equivalent to the remaining lines of $\mathcal{A}'$. The only remaining issue is to argue that the responses generated by $\mathcal{A}'$ to the oracle calls are identically distributed to the responses made by C_{sk_1,pk_2} in the IND_t experiment. This follows by Lemma 5.2 (below) and by the fact that all secret keys are distributed correctly and the re-encryption values Z_1, Z_2, Z_3 are computed correctly. When Γ is not a decision linear instance, then the encryption y is independent of the message m_t and so the probability that $t' = t$ (and therefore the probability that $\mathcal{A}'$ outputs 1) is exactly $1/2$.

Let E_{in} be the event that $\mathcal{A}'$ outputs a 1 when the input Γ is a proper decision linear instance and let E_{out} be the event that $\mathcal{A}'$ outputs a 1 when the input is not an instance. The theorem follows from the following two calculations:

$$\begin{aligned}\Pr[E_{\text{in}}] &= 1/2\big(\Pr\big[D\big(\mathsf{IND}_1(\Pi, \mathcal{A}, k)\big) = 1\big] + \Pr\big[D\big(\mathsf{IND}_0(\Pi, \mathcal{A}, k) = 0\big)\big]\big)\\ &= 1/2\big(\Pr\big[D\big(\mathsf{IND}_1(\Pi, \mathcal{A}, k)\big) = 1\big] + \big(1 - \Pr\big[D\big(\mathsf{IND}_0(\Pi, \mathcal{A}, k) = 1\big)\big]\big)\big)\\ &= 1/2 + \varepsilon/2,\end{aligned}$$

$$\Pr[E_{\text{out}}] = 1/2. \qquad \square$$

5. The Obfuscator for Re-Encryption

In Fig. 2, we describe an obfuscator Obf for the class of re-encryption circuits C_k relative to the encryption scheme Π defined in the previous section.

5.1. *Main Result*

Theorem 5.1. *The obfuscator in Fig. 2 is an average-case secure obfuscator for the circuit family C_k under the Strong Diffie–Hellman Indistinguishability and Decision Linear assumptions in $\mathbb{G}$.*

Algorithm Obf, on input a circuit $C_{sk_1,pk_2} \in C_k$,

1. Reads $sk_1 = (a_1, b_1, g)$ and $pk_2 = (h^{a_2}, h^{b_2}, h)$ from the description of C_{sk_1,pk_2}.
2. Selects a random integer $z \xleftarrow{r} \mathbb{Z}_q^*$ and computes the re-encryption tuple $(Z_1, Z_2, Z_3) = ((h^{a_2})^{z/a_1}, (h^{b_2})^{z/b_1}, h^z)$.
3. Constructs and outputs an obfuscated circuit $R_{pk_1,pk_2,Z}$ that does the following:
   - On input `keys`, output $pk_1 = (g^{a_1}, g^{b_1}, g)$ and $pk_2 = (h^{a_2}, h^{b_2}, h)$.
   - On input a 5-tuple $[0, W, X, Y, 0]$ where $W, X, Y \in \mathbb{G}$:
     (a) Select input re-randomization values $r, s \xleftarrow{r} \mathbb{Z}_q^*$.
     (b) Re-randomize the input as $W' \leftarrow W \cdot (g^{a_1})^r$, $X' \leftarrow X \cdot (g^{b_1})^s$, and $Y' \leftarrow Y \cdot g^{r+s}$.
     (c) Compute $E \leftarrow \mathbf{e}(W', Z_1)$.
     (d) Compute $F \leftarrow \mathbf{e}(X', Z_2)$.
     (e) Compute $G \leftarrow \mathbf{e}(Y', Z_3)$.
     (f) Select an output re-randomization value $y \xleftarrow{r} \mathbb{Z}_q^*$.
     (g) Output the ciphertext $[1,\ E^y,\ F^y,\ G^y,\ Z_3^y]$.
   - Otherwise return $\perp$.

Fig. 2. Obfuscator Obf for Re-encryption circuits for Π.

Proof. Let $pk_1 = (g^{a_1}, g^{b_1}, g)$ and $pk_2 = (h^{a_2}, h^{b_2}, h)$ with matching secret keys $((a_1, b_1, g)$ and (a_2, b_2, h) respectively). The polynomial slowdown property follows by inspection because the obfuscated circuit computes a few bilinear maps and exponentiations. The theorem then follows from Lemmas 5.2 and 5.3.

Lemma 5.2 (Preserving Functionality). *Consider any circuit $C_{sk_1,pk_2} \in C_k$ and let circuit $R_{pk_1,pk_2,Z} \leftarrow \mathsf{Obf}(C_{sk_1,pk_2})$. On every possible input, the output distributions of C_{sk_1,pk_2} and $R_{pk_1,pk_2,Z}$ are identical.*

Proof. We must consider three classes of inputs. First, for a properly formed encryption of any message $m \in M$, observe that

$$\mathsf{Enc}(pk_1, 0, m) = \left[0,\ g^{a_1 r},\ g^{b_1 s}, g^{r+s} \cdot m,\ 0\right]$$

for some $r, s \xleftarrow{r} \mathbb{Z}_q^*$. When such a ciphertext is fed as input to R, the circuit outputs

$$\left[1,\ \mathbf{e}\left(g^{a_1(r+r')}, h^{a_2 z/a_1}\right)^y,\ \mathbf{e}\left(g^{b_1(s+s')}, h^{b_2 z/b_1}\right)^y,\ \mathbf{e}\left(g^{r+s+r'+s'} \cdot m, h^z\right)^y,\ h^{yz}\right]$$

for randomly chosen $r', s', y \xleftarrow{r} \mathbb{Z}_q^*$. Substituting $\overline{r} = \frac{r+r'}{\ell}$, $\overline{s} = \frac{s+s'}{\ell}$, $\overline{t} = h^{yz}$, where ℓ is such that[7] $g^\ell = h$, this 5-tuple can be rewritten as

$$\left[1,\ \mathbf{e}\left(g^{\ell \cdot a_1 \overline{r}}, (h^{yz})^{a_2/a_1}\right),\ \mathbf{e}\left(g^{\ell \cdot b_2 \overline{s}}, (h^{yz})^{b_2/b_1}\right),\ \mathbf{e}\left(g^{\ell \cdot \overline{r}+\overline{s}} \cdot m, \left(h^{yz}\right)\right),\ \overline{t}\right]$$

[7] We do not need to compute ℓ explicitly.

which can be further simplified to

$$\left[1,\ \mathbf{e}\left(h^{a_2\overline{r}}, \overline{t}\right),\ \mathbf{e}\left(h^{b_2\overline{s}}, \overline{t}\right),\ \mathbf{e}\left(h^{\overline{r}+\overline{s}} \cdot m, \overline{t}\right),\ \overline{t}\right]$$

where $\overline{r}, \overline{s}$ are uniformly random elements of $\mathbb{Z}_q^*$, and $\overline{t}$ is a uniformly distributed element of $\mathbb{G}$. This tuple is identically distributed to the output of $\mathsf{Enc}(pk_2, 1, m)$. The same holds for all $m \in \mathbb{G} \setminus M$. For the input `keys` and ill-formed inputs, the outputs are also identical. □

Virtual Blackbox In order to satisfy the virtual black-box property, it suffices to only consider the "dummy" adversary which outputs the code of the obfuscated circuit $\mathsf{Obf}(C)$. Thus, we must construct a simulator $\mathsf{Sim}^C(1^k, z)$ such that for all distinguishers D^C which take as input an obfuscated circuit and auxiliary input z,

$$\left|\Pr\left[D^C\left(\mathsf{Obf}(C), z\right) = 1\right] - \Pr\left[D^C\left(\mathsf{Sim}^C(1^k, z), z\right) = 1\right]\right| < \mu(k)$$

where the probability is taken over the choice of C, and the random coins of Obf, D and Sim.

Define the simulator $\mathsf{Sim}^C(1^k, z)$ as follows:

1. Query the oracle C on `keys` to get pk_1, pk_2.
2. Sample $Z_1', Z_2', Z_3' \xleftarrow{r} \mathbb{G}$.
3. Create and output a circuit $R'_{pk_1, pk_2, Z'}$ using the values $(pk_1, pk_2, Z_1', Z_2', Z_3')$.

Notice that Sim^C produces a circuit which does not correctly compute the re-encryption function. However, we now show that under appropriate complexity assumptions, no p.p.t. distinguisher D^C will notice.

Toward this goal, notice that the output of $D^C(\mathsf{Obf}(C), z)$ is distributed identically to $\mathsf{Nice}(D^C, k, z)$ and the output of $D^C(\mathsf{Sim}^C(1^k, z))$ is distributed identically to $\mathsf{Junk}(D^C, k, z)$ where

$\mathsf{Nice}\left(D^C, k, z\right)$
- $q, \mathbb{G} \leftarrow \mathsf{BMsetup}\left(1^k\right)$
- $g, h, r \xleftarrow{r} \mathbb{G}$
- $a_1, a_2, b_1, b_2 \xleftarrow{r} \mathbb{Z}_q$
- $pk_1 \leftarrow \left(g^{a_1}, g^{b_1}, g\right)$
- $pk_2 \leftarrow \left(h^{a_2}, h^{b_2}, h\right)$
- $Z_1 \leftarrow r^{a_2/a_1};\ Z_2 \leftarrow r^{b_2/b_1}$
- $b \leftarrow D^C(pk_1, pk_2, Z_1, Z_2, r, z)$
- Output b,

$\mathsf{Junk}\left(D^C, k, z\right)$
- $q, \mathbb{G} \leftarrow \mathsf{BMsetup}\left(1^k\right)$
- $g, h, r \xleftarrow{r} \mathbb{G}$
- $a_1, a_2, b_1, b_2 \xleftarrow{r} \mathbb{Z}_q$
- $pk_1 \leftarrow \left(g^{a_1}, g^{b_1}, g\right)$
- $pk_2 \leftarrow \left(h^{a_2}, h^{b_2}, h\right)$
- $Z_1', Z_2' \xleftarrow{r} \mathbb{G}$
- $b \leftarrow D^C(pk_1, pk_2, Z_1', Z_2', r, z)$
- Output b.

In the above experiments, we write $\mathsf{expt}(D^C, k, z)$ to mean that the distinguisher D has oracle access to C_{sk_1, pk_2} for the keys sk_1, pk_2 chosen in the experiment. The virtual blackbox property follows immediately from the following lemma. □

Lemma 5.3. *Under the Strong Diffie–Hellman Indistinguishability and Decision Linear assumptions in* $\mathbb{G}$*, for all p.p.t. distinguishers* D *and auxiliary information* z*, the following two distributions are statistically indistinguishable*:[8]

$$\{\mathsf{Nice}(D^C, k, z)\}_k \quad \textit{and} \quad \{\mathsf{Junk}(D^C, k, z)\}_k.$$

The C in Lemma 5.3 is chosen at random as part of the experiment Nice or Junk, respectively, and the coins of C are freshly and independently generated (i.e., not chosen by D or based on z.)

Proof Outline We prove this lemma in a series of incremental steps. We begin with a simple indistinguishability problem and incrementally add elements and provide access to various oracles until the experiments are equivalent to their final form in Lemma 5.3. Let us now start with a claim which relates the Strong Diffie–Hellman Indistinguishability (SDHI) problem to a simple indistinguishability problem. (In all of the following experiments, we implicitly generate $q, \mathbb{G} \leftarrow \mathsf{BMsetup}(1^k)$ and omit the index k and auxiliary input z when the context is clear.)

Proposition 5.4. *Under the SDHI assumption,* $\mathsf{Nice}^{(1)}_{k,z} \stackrel{c}{\approx} \mathsf{Junk}^{(1)}_{k,z}$ *where*

$\mathsf{Nice}^{(1)}$: *Proceeds as* Nice *except that the output is* $\left(g^{a_1}, g, h^{a_2}, h, Z_1, r, z\right)$,
$\mathsf{Junk}^{(1)}$: *Proceeds as* Junk *except that the output is* $\left(g^{a_1}, g, h^{a_2}, h, Z'_1, r, z\right)$.

If there exists a distinguisher D *which distinguishes* $\mathsf{Nice}^{(1)}$ *from* $\mathsf{Junk}^{(1)}$ *with advantage* ε*, then there exists a distinguisher* D' *which solves the SDHI problem with the same advantage* (*in roughly the same time*).

Proof. The algorithm $D'(g, g^a, g^b, g^c, Q, z)$ works as follows:

1. D' chooses a random $w \stackrel{r}{\leftarrow} \mathbb{Z}_q$.
2. D' runs $D(g^w, (g^b)^w, g^a, g, Q, g^c, z)$ and echoes its output.

Consider $a_1 = 1/b$, $a_2 = a$ and $r = g^c$. Thus, if $Q = g^{abc}$, then we have $Q = r^{ab} = r^{a_2/a_1}$ in which case the input to D is identically distributed to $\mathsf{Nice}^{(1)}$. Otherwise, Q is equal to r^t for some random t and the input to D is identically distributed to $\mathsf{Junk}^{(1)}$. □

We now extend Proposition 5.4 to include more input values.

Proposition 5.5. *Under the SDHI assumption,* $\mathsf{Nice}^{(2)}_{k,z} \stackrel{c}{\approx} \mathsf{Junk}^{(2)}_{k,z}$ *where*

$\mathsf{Nice}^{(2)}$: *Same as* Nice *except that the output is* $(pk_1, pk_2, Z_1, Z_2, r, z)$,
$\mathsf{Junk}^{(2)}$: *Same as* Junk *except that the output is* $(pk_1, pk_2, Z'_1, Z'_2, r, z)$.

Proof. Consider the hybrid distribution $T^{(2)}$ which is the same as $\mathsf{Nice}^{(2)}$ except that $Z'_2 \stackrel{r}{\leftarrow} \mathbb{G}$ and the output is $(pk_1, pk_2, Z_1, Z'_2, r, z)$. If $\mathsf{Nice}^{(2)}$ and $\mathsf{Junk}^{(2)}$ are distinguishable with advantage ε, then either $\mathsf{Nice}^{(2)}$ and $T^{(2)}$ or $T^{(2)}$ and $\mathsf{Junk}^{(2)}$ are distinguishable by algorithm D with advantage $\varepsilon/2$. Either case implies a distinguisher for $\mathsf{Nice}^{(1)}$

[8] The statistical indistinguishability follows because both experiments output a single bit.

from $\mathsf{Junk}^{(1)}$. In the later case, this involves picking $b_1, b_2 \in \mathbb{Z}_q$ to form public keys, picking Z_2' randomly, and using the input instance from $\mathsf{Nice}^{(1)}$ (or $\mathsf{Junk}^{(1)}$) to simulate the input distribution for D. The former case does the same, but swaps the role of a_i and b_i. □

Toward the proof of our main theorem, we now extend Proposition. 5.5 by providing the distinguisher with an oracle which returns a 5-tuple of random values which works as follows. On input $[0, W, X, Y, 0]$, where $W, X, Y \in \mathbb{G}$, $\mathcal{R}$ selects three random values $E, F, G \xleftarrow{r} \mathbb{G}_T$ and a random value $H \xleftarrow{r} \mathbb{G}$ and returns $[1, E, F, G, H]$. Otherwise, $\mathcal{R}$ returns $\perp$. Intuitively, oracle $\mathcal{R}$ outputs only random values and thus should not help any distinguisher.

Proposition 5.6. *Under the SDHI assumption, and for any p.p.t. D, $\mathsf{Nice}^{(3)}_{k,z} \overset{s}{\approx} \mathsf{Junk}^{(3)}_{k,z}$ where*

$\mathsf{Nice}^{(3)}$: *Same as* $\mathsf{Nice}\big(D^{\mathcal{R}}, k, z\big)$,
$\mathsf{Junk}^{(3)}$: *Same as* $\mathsf{Junk}\big(D^{\mathcal{R}}, k, z\big)$.
(*That is, the distinguishers have oracle access to $\mathcal{R}$ and output a bit instead of a tuple as in the* $^{(2)}$*-experiments.*)

Proof. The oracle $\mathcal{R}$ can be perfectly simulated without any auxiliary information. Thus, for any D^R, there exists another non-oracle distinguisher D' (which internally runs D while perfectly simulating $\mathcal{R}$ to D) whose output distribution is identical to D. Applying Proposition 5.5, we thus have that for all distinguishers D^R, $\mathsf{Nice}^{(2)} \overset{c}{\approx} \mathsf{Junk}^{(2)}$ which implies $\mathsf{Nice}^{(3)} \overset{s}{\approx} \mathsf{Junk}^{(3)}$ (since the later experiment outputs a bit). □

We now consider the distinguisher with oracle access to the real re-encryption circuit C.

Proposition 5.7. *For any p.p.t. distinguisher D^C, let*

$$\alpha(k, z) = \Pr\big[\mathsf{Nice}\big(D^C, k, z\big) = 1\big] - \Pr\big[\mathsf{Junk}\big(D^C, k, z\big) = 1\big],$$
$$\beta(k, z) = \Pr\big[\mathsf{Nice}\big(D^{\mathcal{R}}, k, z\big) = 1\big] - \Pr\big[\mathsf{Junk}\big(D^{\mathcal{R}}, k, z\big) = 1\big].$$

There exists a p.p.t. algorithm $\mathcal{A}$ which distinguishes between the two distributions of the decision linear problem with advantage at least $\frac{1}{2}|(\alpha(k, z) - \beta(k, z))|$.

Proof. We take $\alpha = \alpha(k, z)$ and $\beta = \beta(k, z)$. The algorithm $\mathcal{A}$ takes as input, a decision linear instance $\Gamma = (h_1, h_2, h, h_1^x, h_2^y, Q)$ and auxiliary information z, and:

1. $\mathcal{A}$ samples a challenge bit $c \xleftarrow{r} \{0, 1\}$ to pick whether to run Nice or Junk.
2. $\mathcal{A}$ samples integers $a, b, u \xleftarrow{r} \mathbb{Z}_q$ and group elements $g, Z_1', Z_2', Z_3' \xleftarrow{r} \mathbb{G}$.
3. $\mathcal{A}$ sets $pk_1 = (g^a, g^b, g)$ and $pk_2 = (h_1, h_2, h)$ and computes a valid re-encryption tuple (Z_1, Z_2, Z_3) by $Z_1 \leftarrow h_1^{u/a}$, $Z_2 \leftarrow h_2^{u/b}$, and $Z_3 \leftarrow h^u$.

4. If $c = 1$, then $\mathcal{A}$ runs $D^{\mathcal{O}}(pk_1, pk_2, Z_1, Z_2, Z_3, z)$ where $\mathcal{O}$ is defined below.
   If $c = 0$, then $\mathcal{A}$ runs $D^{\mathcal{O}}(pk_1, pk_2, Z'_1, Z'_2, Z'_3, z)$.
   When D queries the oracle $\mathcal{O}$ on input $[0, W, X, Y, 0]$, $\mathcal{A}$ responds as follows:
   (a) Sample input re-randomization values $r, s, t \xleftarrow{r} \mathbb{Z}_q$.
   (b) Re-randomize the input as $W' \leftarrow W \cdot g^{ar}$, $X' \leftarrow X \cdot g^{bs}$, and $Y' \leftarrow Y \cdot g^{r+s}$.
   (c) Compute $E \leftarrow \mathbf{e}(W', Z_1) \cdot \mathbf{e}(g, h_1^{tx})$.
   (d) Compute $F \leftarrow \mathbf{e}(X', Z_2) \cdot \mathbf{e}(g, h_2^{ty})$.
   (e) Compute $G \leftarrow \mathbf{e}(Y', Z_3) \cdot \mathbf{e}(g, Q^t)$.
   (f) Sample output re-randomization value $\ell \xleftarrow{r} \mathbb{Z}_q$.
   (g) Respond with the ciphertext $[1, E^\ell, F^\ell, G^\ell, Z_3^\ell]$.
   Whenever D queries its oracle on input `keys`, $\mathcal{A}$ responds with pk_1 and pk_2, and on all other queries, $\mathcal{A}$ responds with $\perp$.
5. Eventually D outputs $c' \in \{0, 1\}$. If $c = c'$, $\mathcal{A}$ outputs 1 (i.e., it guesses that $Q = h^{x+y}$). Else if $c \neq c'$, then $\mathcal{A}$ outputs 0 (i.e., it guesses that $Q \neq h^{x+y}$).

Note that when $\mathcal{A}$ responds to queries made to $\mathcal{O}$, it almost mimics the real obfuscated program (C). The difference is that when computing (4)–(4), additional terms are multiplied with elements of the ciphertext. If $Q = h^{x+y}$, i.e. the instance Γ instance is a decision linear tuple, then these operations simply contribute to additional re-randomization of the ciphertext that does not change the ciphertext distribution. However, if Q is a random value, i.e. Γ is not a decision linear instance, then these operations make E, F, G a random 3-tuple that is also independent of Z_3. This idea is the essential step in the proof formalized in the two claims below.

Claim. *When $Q = h^{x+y}$ (i.e. Γ is a decision linear instance), then* $\Pr[\mathcal{A}(\Gamma) = 1] = \frac{1}{2} + \alpha(k, z)/2$.

Proof of Claim. When $Q = h^{x+y}$, then $\mathcal{A}$ perfectly simulates Nice^C or Junk^C toward the algorithm D. The key point is to recognize that (h_1, h_2, h) can be interpreted as a randomly generated public key since h_1, h_2 can be rewritten as $h_1 = h^{e_1}$ and $h_2 = h^{e_2}$ for some (unknown) e_1, e_2. Since the re-encryption tuple Z_1, Z_2, Z_3 is also a valid re-encryption tuple for $pk_1 \to pk_2$, the input parameters to D in step 4 are identically distributed to the inputs to D in either experiment Nice or Junk. Moreover, the response to an oracle query on `keys` is also identically distributed. All that remains is to show that the responses $\mathcal{A}$ provides to oracle queries on $[0, W, X, Y, 0]$ are also identically distributed. This last point follows by inspection because $Q = h^{x+y}$ and Z_1, Z_2, Z_3 are a valid re-encryption tuple. A simple probability analysis completes the result:

$$\begin{aligned}
\Pr\big[\mathcal{A}(\Gamma) = 1 \mid \Gamma \in DL\big] &= \frac{1}{2}\big(\Pr\big[\mathsf{Nice}\big(D^C\big) = 1\big] + \Pr\big[\mathsf{Junk}\big(D^C\big) = 0\big]\big) \\
&= \frac{1}{2}\big(\Pr\big[\mathsf{Nice}\big(D^C\big) = 1\big] + 1 - \Pr\big[\mathsf{Junk}\big(D^C\big) = 1\big]\big) \\
&= \frac{1}{2} + \frac{\Pr[\mathsf{Nice}(D^C) = 1] - \Pr[\mathsf{Junk}(D^C) = 1]}{2} \\
&= \frac{1}{2} + \frac{\alpha}{2}.
\end{aligned}$$

□

Claim. *If Q is randomly chosen value, then* $\Pr[\mathcal{A}(\Gamma) = 1] = \frac{1}{2} + \beta(k, z)/2$.

Proof of Claim. This proof is almost identical to the previous one. The only difference is we must show that responses to the oracle queries return four randomly selected group elements. Let us denote by $\omega, \chi, \gamma, \upsilon$ the values such that $W = g^{\omega}$, $X = g^{\chi}$, $Y = g^{\gamma}$ and $Q = h^{\upsilon}$, and by e_1, e_2 the values such that $h_1 = h^{e_1}$ and $h_2 = h^{e_2}$. Observe that the elements returned by the oracle are

$$E = \left[\mathbf{e}\left(W \cdot g^{ar}, h_1^{u/a}\right) \cdot \mathbf{e}\left(g, h_1^{tx}\right)\right]^{\ell} = \mathbf{e}(g, h)^{\ell e_1[\omega u/a + tx] + ru\ell e_1},$$

$$F = \left[\mathbf{e}\left(X \cdot g^{bs}, h_2^{u/b}\right) \cdot \mathbf{e}\left(g, h_2^{ty}\right)\right]^{\ell} = \mathbf{e}(g, h)^{\ell e_2[\chi u/b + ty] + su\ell e_2},$$

$$G = \left[\mathbf{e}\left(Y \cdot g^{r+s}, h^{u}\right) \cdot \mathbf{e}\left(g, Q^{t}\right)\right]^{\ell} = \mathbf{e}(g, h)^{\ell[(\gamma + r + s)u] + t\upsilon\ell},$$

$$H = h^{u\ell}.$$

Since r, s, t, ℓ are fresh independently selected values, then E, F, G, H will also be independent on every invocation of the oracle. This follows because each of E, F, G, H can be viewed as either a generator of a group which has been raised to a random power, or as an element of the group which has been multiplied by a random group element. □

To complete the proof of Proposition 5.7, notice that the two claims above imply that when Γ is a decision linear instance, then $\mathcal{A}$ outputs 1 with probability $\frac{1}{2} + \frac{\alpha}{2}$, whereas when Γ is not a decision linear instance $\mathcal{A}$ outputs 1 with probability $\frac{1}{2} + \frac{\beta}{2}$. Thus, $\mathcal{A}$ distinguishes the two distributions of the decision linear problem with advantage at least $\frac{1}{2}|(\alpha(k, z) - \beta(k, z))|$. □

Proof of Lemma 5.3. By the decision linear assumption and Proposition 5.7, it follows that $|\alpha(k, z) - \beta(k, z)|$ is negligible. By Proposition 5.6, $\beta(k, z)$ must be a negligible function, and therefore, so too must $\alpha(k, z)$. This establishes the lemma. □

6. Conclusions

We presented a positive obfuscation result for a natural cryptographic functionality. Our focus is on the *re-encryption* functionality, which takes a ciphertext for message m encrypted under Alice's public key and transforms it into a ciphertext for the same message m under Bob's public key.

We considered a definition of obfuscation for probabilistic circuits that focuses on *average-case* security and provides a *composable* guarantee. In Theorem 2.3, we argued that this last property captures the intuition that: *If a cryptographic scheme is* "*secure*" *when the adversary is given black-box access to a program, then it remains* "*secure*" *when the adversary is given the obfuscated version of the program.*

In Definition 4.3, we formalized what it means for an encryption algorithm to be "secure" when the adversary is given black-box access to a re-encryption oracle. We observe that this security experiment can be simulated in polynomial time using access to the re-encryption oracle. We presented an encryption scheme in Fig. 1 and showed in

Theorem 4.5 that is satisfies this definition. Finally, we presented a composable obfuscator for the re-encryption functionality in Fig. 2 and proved that it met Definition 2.2 in Theorem 5.1. Given Theorem 2.3, this collection of results allow us to conclude that:

Corollary 6.1. *Under the Strong Diffie–Hellman Indistinguishability and Decision Linear assumptions in* $\mathbb{G}$, *the encryption scheme in Fig. 1 is indistinguishable secure when oracle* C *is replaced by access to* $\mathsf{Obf}(C)$ *as defined in Fig. 2.*

In several practical applications, it is desirable to replace black-box access to a program with an obfuscation of the program in a manner that provably does not undermine the security of other cryptographic systems in the environment. For one cryptographic functionality, we have shown that this is possible.

Acknowledgements

We are grateful to Ran Canetti for suggesting the problem and to Shafi Goldwasser for informative conversations. We thank Rafael Pass for illuminating discussions and for suggesting the framework of Definition 2.2. Thanks also to the anonymous TCC 2007 and Journal of Cryptology reviewers for their helpful comments.

References

[1] G. Ateniese, K. Fu, M. Green, S. Hohenberger, Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.* **9**(1), 1–30 (2006)

[2] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S.P. Vadhan, K. Yang, On the (im)possibility of obfuscating programs, in *Advances in Cryptology—CRYPTO'01*, ed. by J. Kilian. Lecture Notes in Computer Science, vol. 2139 (Springer, Berlin, 2001), pp. 1–18

[3] M. Blaze, M. Strauss, Atomic proxy cryptography. Technical report, AT&T Research (1997)

[4] M. Blaze, G. Bleumer, M. Strauss, Divertible protocols and atomic proxy cryptography, in *Advances in Cryptology—EUROCRYPT'98*, ed. by K. Nyberg. Lecture Notes in Computer Science, vol. 1403 (Springer, Berlin, 1998), pp. 127–144

[5] D. Boneh, X. Boyen, H. Shacham, Short group signatures, in *Advances in Cryptology—CRYPTO'04*, ed. by M.K. Franklin. Lecture Notes in Computer Science, vol. 3152 (Springer, Berlin, 2004), pp. 41–55

[6] D. Boneh, A. Sahai, B. Waters, Fully collusion resistant traitor tracing with short ciphertexts and private keys, in *Advances in Cryptology—EUROCRYPT'06*, ed. by S. Vaudenay. Lecture Notes in Computer Science, vol. 4004 (Springer, Berlin, 2006), pp. 573–592

[7] R. Canetti, Towards realizing random oracles: Hash functions that hide all partial information, in *Advances in Cryptology—CRYPTO'97*, ed. by B.S. Kaliski Jr. Lecture Notes in Computer Science, vol. 1294 (Springer, Berlin, 1997), pp. 455–469

[8] R. Canetti, D. Micciancio, O. Reingold, Perfectly one-way probabilistic hash functions (preliminary version), in *30th Symposium on Theory of Computing (STOC)* (ACM, New York, 1998), pp. 131–140

[9] Y. Dodis, A. Ivan, Proxy cryptography revisited, in *10th Network and Distributed System Security Symposium (NDSS)*, ed. by V. Gligor, M. Reiter. The Internet Society (2003)

[10] Y. Dodis, A. Smith, Correcting errors without leaking partial information, in *37th ACM Symposium on Theory of Computing (STOC)*, ed. by H.N. Gabow, R. Fagin. (ACM, New York, 2005), pp. 654–663

[11] S.D. Galbraith, K.G. Paterson, N.P. Smart, Pairings for cryptographers. Cryptology ePrint Archive: Report 2006/165 (2006)

[12] S. Goldwasser, Y.T. Kalai, On the impossibility of obfuscation with auxiliary input, in *46th IEEE Symposium on Foundations of Computer Science (FOCS)*, ed. by É. Tardos. (IEEE Comput. Soc., Los Alamitos, 2005), pp. 553–562

[13] S. Goldwasser, S. Micali, Probabilistic encryption. *J. Comput. Syst. Sci.* **28**(2), 270–299 (1984)
[14] S. Hada, Zero-knowledge and code obfuscation, in *Advances in Cryptology—ASIACRYPT'00*, ed. by T. Okamoto. Lecture Notes in Computer Science, vol. 1976 (Springer, Berlin, 2000), pp. 443–457
[15] D. Hofheinz, J. Malone-Lee, M. Stam, Obfuscation for cryptographic purposes, in *4th Theory of Cryptography Conference (TCC)*, ed. by S.P. Vadhan. Lecture Notes in Computer Science, vol. 4392 (Springer, Berlin, 2007), pp. 214–232
[16] M. Mambo, E. Okamoto, Proxy cryptosystems: Delegation of the power to decrypt ciphertexts. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **E80-A**(1), 54–63 (1997)
[17] R. Pass, Personal Communication (2006)
[18] T. Smith, DVD Jon: buy DRM-less Tracks from Apple iTunes, 18 March, 2005. http://www.theregister.co.uk/2005/03/18/itunes_pymusique
[19] L.G. Valiant, A theory of the learnable. *Commun. ACM* **27**(11), 1134–1142 (1984)
[20] H. Wee, On obfuscating point functions, in *37th ACM Symposium on Theory of Computing (STOC)*, ed. by H.N. Gabow, R. Fagin. (ACM, New York, 2005), pp. 523–532