Obfuscation $\Rightarrow$ (IND-CPA Security $\not\Rightarrow$ Circular Security)

(Draft Version, 24 October 2013)[⋆]

Antonio Marcedone[1,⋆⋆] and Claudio Orlandi[2]

[1] Scuola Superiore di Catania, University of Catania, Italy, amarcedone@cs.au.dk
[2] Aarhus University, Denmark, orlandi@cs.au.dk

Abstract *Circular security* is an important notion for public-key encryption schemes and is needed by several cryptographic protocols. In circular security the adversary is given an extra "hint" consisting of a *cycle* of encryption of secret keys i.e., $(E_{pk_1}(sk_2), \ldots, E_{pk_n}(sk_1))$. A natural question is whether every IND-CPA encryption scheme is also circular secure. It is trivial to see that this is not the case when $n = 1$. In 2010 a separation for $n = 2$ was shown by [ABBC10, GH10] under standard assumptions in bilinear groups.
In this paper we finally settle the question showing that for every n there exist an IND-CPA secure scheme which is not n-circular secure. Our result relies on the recent progress in program obfuscation.

1 Introduction

Public-key encryption schemes allow to take any plaintext and create a corresponding ciphertext that carries little or no information about the encrypted plaintext, in the eyes of everyone else but the owner of the secret key.

One might think that for an encryption scheme all plaintexts are equal, but it turns out that some plaintexts are more equal than others. In particular, secret-keys (or functions of them) are a very special kind of plaintexts.

But why would you want to encrypt a secret key? A prime example is fully-homomorphic encryption (FHE): At the heart of virtually every fully-homomorphic encryption schemes there is a technique called "bootstrapping" that requires users to publish, in their public key, an encryption of the secret key [Gen09]. For another example think of two cryptographers, Alice and Bob, who get married and decide they should not keep any secret from each other and therefore decide to share their secret keys with each other. To do so Alice sends an encryption of her secret key sk_A to Bob using his public key pk_B, while Bob sends an encryption of his secret key sk_B to Alice using her public key pk_A. This is not a far fetched example and there are applications where this is needed, see [CL01].

Suppose now that the the evil eavesdropper Eve gets to see these encryption of secret keys: is the encryption scheme still secure, or can Eve use this extra information to break its security?

Circular Security. In the FHE example, a secret key was encrypted under its own secret key and we call this a 1-cycle i.e., Eve learns $E_{pk}(sk)$. When Alice and Bob both encrypt their secret keys under the other party public key, we get a 2-cycle i.e., Eve learns $E_{pk_A}(sk_B)$ and $E_{pk_B}(sk_A)$. In general, we are interested in what happens when Eve learns the encryptions of n secret keys $(sk_1, \ldots, sk_n)$ under public keys $(pk_2, \ldots, pk_n, pk_1)$ respectively. If an encryption scheme is still secure when the adversary is given such a cycle of encryptions

⋆ This is a draft version of a work in progress that we felt compelled to publish due to the fact that an independent work showing a very similar result appeared today on ePrint [KRW13].
⋆⋆ Work done while visiting Aarhus University.

of secret keys, we say that the scheme is n-circular secure[3]. This notion was firstly defined in [CL01, BRS02]. Since then it has been an open problem to understand the relationship between the standard definition of security for public key encryption schemes (namely *indistinguishability under chosen-plaintext-attack* or *IND-CPA* for short) and n-circular security.

IND-CPA Security $\not\Rightarrow$ 1-Circular Security. It is quite easy to show that IND-CPA security does not imply 1-circular security. Take any IND-CPA secure scheme (G, E, D) and construct (G, E', D) as follows: On input m, the modified encryption scheme $E'_{pk}(\cdot)$ first checks if $m \stackrel{?}{=} sk$.[4] If so, E' outputs m, else it outputs $E_{pk}(m)$. The modified scheme is still IND-CPA secure (as it behaves exactly like E for all $m \neq sk$), but since $E'_{pk}(sk) = sk$ it is clear that it would be a very bad idea to let Eve learn this value.

Pairing Assumptions $\Rightarrow$ (IND-CPA Security $\not\Rightarrow$ 2-Circular Security). Surprisingly, it was quite harder to show that there are IND-CPA schemes that are not 2-circular secure. The reason for this is that the secret keys are generated independently and therefore the encryption algorithm does not have a way of distinguishing a secret key from a message (in fact, every message could be a secret key). This problem has been open for about a decade until it was finally solved in 2010 by [ABBC10, GH10]. Both these results hold under the assumptions that some problems are hard in bilinear groups. The counterexample is obtained by embedding some extra elements in the ciphertexts. These extra values in the ciphertext that do not help the adversary to break the IND-CPA game but, when combined together using a bilinear maps, allow to effectively decrypt one of the two ciphertexts and recover a secret key.

Obfuscation $\Rightarrow$ (IND-CPA Security $\not\Rightarrow$ Circular Security). In this paper, we show that IND-CPA security does not imply n-circular secure *for any* n. More precisely, for every n, we can construct a scheme that is not n'-circular secure for every $n' < n$. Our result is based on virtual black-box (VBB) obfuscation, as defined by [BGI$^+$01, BGI$^+$12].

(False $\Rightarrow$ True)? One might now object that our theorem is trivial: the same paper that defined VBB obfuscation also proved that this notion is *impossible to achieve!* At least this has been a common misconception for the last decade, but [BGI$^+$01, BGI$^+$12] "only" proved that there exist no single obfuscator that can obfuscate every program, and their result does not rule out the existence of specific obfuscators for specific class of functions, such as [CD08, Wee05, CRV10, HRSV11].

In a surprising turn of event – and thanks to the recent breakthrough on a candidate for multilinear maps [GGH13a] – the first candidate cryptographic obfuscation was presented in [GGH$^+$13b]. The obfuscation of [GGH$^+$13b] does not contradict the impossibility result of [BGI$^+$01, BGI$^+$12], as it achieves a weaker notion called *indistinguishability obfuscation*. Yet, this weaker notion seems to be sufficient for most practical purposes as someone has commented this result as "*an impractical obfuscation for all practical purposes*"[5].

In this paper, we will make use of the recent candidate VBB obfuscation from [BR13]. This result overcomes the impossibility result of [BGI$^+$01, BGI$^+$12], by proving the security of the scheme in the *generic graded encoding scheme model*: this can be thought as the analogue of the *generic group model* for *discrete logarithm*, extended to the case of multilinear maps. This is arguably a very strong model. However, thanks to VBB obfuscation we can present our result in a very clear and conceptually elegant way.

We conjecture that our result holds also in the standard model (using the weaker notion of indistinguishability obfuscation), and we are working on a proof to appear in the next version of this preprint.

[3] There are different ways of defining circular security. The interested reader can check [CGH12] and reference therein for a discussion of the definitions. In this paper we will show a scheme where the adversary (given a cycle of encryption of secret keys) can recover *all the secret keys*, thus breaking even the weakest notions of circular security. Therefore the actual definition used is irrelevant for us.

[4] Note that it is always possible to check if $m = sk$ by, for example, encrypting a bunch of random messages using $E_{pk}(\cdot)$ and decrypting them using m i.e., the encryption algorithm checks if $D_m(E_{pk}(r)) = r$ for enough random values r. If the results are all correct, one can assume whp that $m = sk$.

[5] Cit. Yuval Ishai.

The good news. While our work provides strong evidence for the fact that not all IND-CPA secure public key encryption schemes achieve circular security, there are a number of encryption schemes that can be proven secure even under these attacks. We refer the interested reader to [BHHO08, CGH12, Hof13, BGK11, BG10] and references therein.

1.1 Technical Overview

The simplest way of constructing a public-key encryption scheme in a world where obfuscation exists is probably the follow: a secret key is just a random string s and a public key is a program P that outputs 1 on input s and $\perp$ otherwise. We write $s \xrightarrow{P} 1$ for compactness. We can think of a plaintext m as a program $1 \to m$. Now to encrypt m under public key P one can construct a ciphertext C with the functionality $s \xrightarrow{C} m$ by composing the two programs $s \xrightarrow{P} 1 \to m$. Correctness is trivial to check and security follows from the fact that the programs are obfuscated and can therefore only be used as "black-boxes".

To break the security of the scheme, we need to add some other information to the public keys that "recognizes" circular encryptions without otherwise affecting the security of our scheme. We do so by appending an obfuscated program Q to the public key. The program Q outputs other obfuscated programs who remember the secret key s. When creating a ciphertext we append this new program R, obtained by calling Q on input m to the ciphertext. The program R now remembers both m and s and accepts as input other programs (aka ciphertexts).

The program R provides the following functionality: On input a ciphertext C, the program R tries to decrypt C with m and, if the output is s, releases the secret key s. So, if Q_1 is the program that remembers s_1 and then it is invoked on a message s_2, the ciphertext will now contain a program $R_{1,2}$ that can recognize encryptions of s_1 under the key s_2.

The next observation is that any two programs $x \xrightarrow{A} y$ and $y \xrightarrow{B} z$ can be composed into a program $x \xrightarrow{C} z$. In particular, from a set of n encryptions $s_i \xrightarrow{C_i} s_{(i+1 \mod n)}$ for $i = 1, \ldots, n$ one can compute n programs

$$s_{(i+1 \mod n)} \xrightarrow{C_i^*} s_i$$

Clearly the size of these programs grows with n, but this is not a problem as long as we set the input size of R to be big enough.

This concludes the intuitive description of our "attack". In the next sections we will provide a formal explanation of this intuitive idea and prove that appending Q to the public key and R to the ciphertext does not affect the IND-CPA security of our scheme.

2 Separating IND-CPA Security from Circular Security

2.1 Preliminaries

In this section, we state the notation and conventions used in the rest of the work. To keep the paper self contained, we will also recall some relevant definitions and theorems.

Notation and Conventions: We use lowercase letters s, x, y for strings in $\{0,1\}^n$. We use uppercase letters P, C, Q, R for "plaintext" programs and $\overline{P}, \overline{C}, \overline{Q}, \overline{R}$ for obfuscated programs. We call $\mathcal{P}$ the set of all polynomial-time programs. We use the notation $P(x \in X) \in Y$ when we want to say that a program P takes input from $X \cup \{\perp\}$ and returns a value in $Y \cup \{\perp\}$. For all programs we define $P(\perp) = \perp$. We write P^Q when P is a program that has "oracle access" to the function/program Q, and denote by LR the function that on input a bit b and two strings m_0, m_1 returns m_b.

If S is a set $s \leftarrow S$ is a uniformly random sample from S. If A is a randomized algorithm, $x \leftarrow A$ is the output of A on a uniformly random input tape.

Definition 1. *We say that a public key encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* secure against a chosen plaintext attack *if for any* λ*, any PPT adversary* $\mathcal{A}$*, and any couple of messages* x, y *of (the same) length polynomial in* λ*:*

$$\left|\Pr\left[\mathcal{A}(pk, \mathsf{Enc}_{pk}(x)) = 1 \mid (pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)\right] - \Pr\left[\mathcal{A}(pk, \mathsf{Enc}_{pk}(y)) = 1 \mid (pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)\right]\right| \leq \mathrm{negl}(\lambda).$$

VBB Obfuscation: We recall the definition of VBB obfuscation and the theorem stating the assumptions needed to achieve it.

Definition 2 (Virtual Black-Box Obfuscator [BGI+01, BGI+12]). *Let* $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ *be a family of polynomial-size circuits, where* $\mathcal{C}_n$ *is a set of boolean circuits operating on inputs of length* n*. And let* $\mathcal{O}$ *be a PPT algorithm, which takes as input an input a length* $n \in \mathbb{N}$*, a circuit* $C \in \mathcal{C}_n$*, a security parameter* $\lambda \in \mathbb{N}$*, and outputs a boolean circuit* $\mathcal{O}(C)$ *(not necessarily in* $\mathcal{C}$*).*

$\mathcal{O}$ is a (black-box) obfuscator for the circuit family $\mathcal{C}$ if it satisfies:

Preserving Functionality: *For every* $n \in \mathbb{N}$*, every* $C \in \mathcal{C}$ *and every* $\boldsymbol{x} \in \{0,1\}^n$*, with all but* $\mathrm{negl}(\lambda)$ *probability over the coins of* $\mathcal{O}$*:*

$$\left(\mathcal{O}(C, 1^n, \lambda)\right)(\boldsymbol{x}) = C(\boldsymbol{x})$$

Polynomial Slowdown: *For every* $n, \lambda \in \mathbb{N}$ *and* $C \in \mathcal{C}$*, the circuit* $\mathcal{O}(C, 1^n, 1^\lambda)$ *is of size at most* $\mathrm{poly}(|C|, n, \lambda)$*.*

Virtual Black-Box: *For every (non-uniform) polynomial size adversary* $\mathcal{A}$*, there exists a (non-uniform) polynomial size simulator* $\mathcal{S}$*, such that for every* $n \in \mathbb{N}$ *and for every* $C \in \mathcal{C}$*:*

$$\left|\Pr_{\mathcal{O},\mathcal{A}}\left[\mathcal{A}\left(\mathcal{O}(C, 1^n, 1^\lambda)\right) = 1\right] - \Pr_{\mathcal{S}}\left[\mathcal{S}^C(1^{|C|}, 1^n, 1^\lambda) = 1\right]\right| \leq \mathrm{negl}(\lambda)$$

Theorem 1 (Candidate VBB Obfuscation [BR13]). *There exist an obfuscator* $\mathcal{O}$ *for any circuit in* $\mathcal{P}$*, which is virtual black-box secure in the generic graded encoding scheme model, assuming the bounded speedup hypothesis, and the existence of homomorphic encryption with an* NC^1 *decryption circuit.*

2.2 PKE from Obfuscation

We start by constructing a very simple IND-CPA public-key encryption scheme $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ based on obfuscation, and show some of its interesting property. In the next subsection, we will modify it in order to render it insecure under n-circular security attacks.

Key Generation: The algorithm $\mathsf{Gen}(1^k)$ chooses a random secret key $s \leftarrow \{0,1\}^k$. The public key is an obfuscated program $\overline{P} \leftarrow \mathcal{O}(P)$ where P is defined as follows:
def $P(x \in \{0,1\}^k) \in \{0,1\}$:
1. if $(x \stackrel{?}{=} s)$ output 1; else output 0.

Encryption: The algorithm $\mathsf{Enc}(\overline{P}, m)$ on input a public key $\overline{P} \in \mathcal{P}$ and a message $m \in \{0,1\}^k$ outputs an obfuscated program $\overline{C} \leftarrow \mathcal{O}(C)$ where C is defined as follows:
def $C(x \in \{0,1\}^k) \in \{0,1\}^k$:
1. if $(\overline{P}(x) \stackrel{?}{=} 1)$ output m; else output $\perp$.

Decryption: The algorithm $\mathsf{Dec}(s, \overline{C})$ on input a secret key $s \in \{0,1\}^k$ and a ciphertext $\overline{C} \in \mathcal{P}$ outputs $m' = \overline{C}(s)$.

It is easy to check that if $(s, \overline{P}) \leftarrow \mathsf{Gen}$, then:

$$\mathsf{Dec}(s, \mathsf{Enc}(\overline{P}, m)) = m$$

Theorem 2. *If $\mathcal{O}$ is a VBB obfuscator for $\mathcal{P}$ according to Definition 2, then the scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *described above is IND-CPA secure according to Definition 1.*

Proof. The adversary $\mathcal{A}$ for Definition 1 and the adversary for Definition 2 are syntactically equivalent. Let $(\overline{P}, \overline{C_x})$ be an encryption of x and $(\overline{P}, \overline{C_y})$ and encryption of y. Assume that there exist an $\mathcal{A}$ s.t.,

$$|\Pr[\mathcal{A}(\overline{P}, \overline{C_x}) = 1] - \Pr[\mathcal{A}(\overline{P}, \overline{C_y}) = 1]| = \delta > \mathrm{negl}(k)$$

then we will show that $\mathcal{A}$ is such that, for all $\mathcal{S}$, it cannot be that both $\gamma_x = |\Pr[\mathcal{A}(\overline{P}, \overline{C_x}) = 1] - \Pr[\mathcal{S}^{P,C_x} = 1]| = \mathrm{negl}(k)$ and $\gamma_y = |\Pr[\mathcal{A}(\overline{P}, \overline{C_y}) = 1] - \Pr[\mathcal{S}^{P,C_y} = 1]| = \mathrm{negl}(k)$.

Using the triangular inequality we have that, if $\gamma_x + \gamma_y = \mathrm{negl}(k)$ then also:

$$\left|\delta + \Pr[\mathcal{S}^{P,C_x} = 1] - \Pr[\mathcal{S}^{P,C_x} = 1]\right| = \mathrm{negl}(k)$$

But this implies that $\Pr[\mathcal{S}^{P,C_x} = 1] - \Pr[\mathcal{S}^{P,C_x} = 1] > \mathrm{negl}(k)$, and in the ideal world where the simulator has only access to the public key and the ciphertexts as oracle, this happens only if the simulator queries its oracles on the secret key s which happens with probability $\mathrm{poly}(k)2^{-k} = \mathrm{negl}(k)$ and thus we reach a contradiction.

□

2.3 Properties of Our Scheme

The scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ defined in the previous section has an interesting property, namely that it is possible to combine ciphertexts together in order to achieve some flavour of *proxy re-encryption*, namely it is possible to delegate to someone the power to transform ciphertexts encrypted under a public key $\overline{P_1}$ into ciphertexts encrypted under a different public key $\overline{P_2}$ without having to release the corresponding secret key s_1.

To see how this is possible, think of a proxy who is given two public keys $(\overline{P_1}, \overline{P_2})$ and

$$\overline{C_{1\to 2}} = \mathsf{Enc}(\overline{P_2}, s_1)$$

(i.e., an encryption of secret key 2 using public key 1). It will be convenient now to say that a program $\overline{C}$ (not necessarily an output of Enc) is an encryption of m under key i if $\mathsf{Dec}(s_i, \overline{C}) = m$.

Then the proxy, using $\overline{C_1}$ s.t. $\mathsf{Dec}(s_1, \overline{C_1}) = m$ and $\overline{C_{1\to 2}}$ s.t. $\mathsf{Dec}(s_2, \overline{C_{1\to 2}}) = s_1$, can compute an encryption of m under key $\overline{P_2}$ by creating an obfuscated program $\overline{C_2} \leftarrow \mathcal{O}(C_2)$ where C_2 is defined as follows:

def $C_2(x \in \{0,1\}^k) \in \{0,1\}^k$
1. Output $\overline{C_1}(\overline{C_{1\to 2}}(x))$;

It is now easy to check that $\overline{C_2}(s_2) = m$ and that, due to the property of the VBB obfuscator $\mathcal{O}$, nothing else can be computed from $\overline{C_2}$.

***n*-circular security ⇔ 2-circular security:** Using this property, we can go from a cycle of n encryptions to $n-1$ cycles of length 2. Namely, let $\overline{C}_{i\to(i+1)} = \mathsf{Enc}(\overline{P_i}, s_{i+1})$ for all $i \in \{1, \ldots, n\}$ (where all additions are modulo n). Then one can create programs

$$C^*_{(i+1)\to i} = \overline{C}_{(i+1)\to(i+2)} \circ \ldots \circ \overline{C}_{(i-1)\to i}$$

Note that in this case we are not even interested in re-obfuscating the concatenation of the programs (like in the proxy re-encryption application) and the program $C^*_{(i+1)\to i}$ is a "functional ciphertext" in the sense that it is a program which decrypts to s_i on input s_{i+1}. The only difference between C^* and "regular" ciphertext is that the size of C^* grows with n. Given an obfuscator $\mathcal{O}$, it is possible to find an upper bound $\beta_n = \mathrm{poly}(\lambda, n)$ s.t., the size of $C^*_{(i+1)\to i}$ is less than β.

2.4 A PKE that is not n-Circular Secure

We add some elements to the public key and the ciphertext to make the scheme from the previous section insecure under circular attacks:

Key Generation: The secret key is s as above. The public key contains $\overline{P}$ as above. In addition, we append and obfuscated program $\overline{Q}$ to the public key. Let $\mathcal{B}$ be the set of all programs of size at most β_n as defined above, then the program Q defined as follows.
def $Q(y \in \{0,1\}^k) \in \mathcal{P}$:
1. Output an obfuscated program $\overline{R}$ defined as follows:
   def $R(B \in \mathcal{B}) \in \{0,1\}^k$:
   (a) If $(B(y) = s)$, output s; else output $\perp$.

Encryption: An encryption of m now is a pair $(\overline{C}, \overline{R})$ where $\overline{C}$ is defined as above and $\overline{R} \leftarrow \overline{Q}(m)$.

Decryption: Unchanged.

Circular (in)Security of Our Scheme: The only thing we modified is that now an encryption contains a program R that accepts as inputs any ciphertext and tries to decrypt it using the message and checks whether this gives the secret key. In that case, it outputs the secret key.

Then the attack is as follows: let s_1, s_2 be two secret keys and $(\overline{P}_1, \overline{Q}_1), (\overline{P}_2, \overline{Q}_2)$ their respective public keys. The output of $\mathsf{Enc}(\overline{P}_1, s_2)$ is $(\overline{C}_1, \overline{R}_{1,2})$. That is, $\overline{R}_{1,2}$ is a program that accepts as input any program C of size at most β_n, and if $C(s_2) = s_1$ it outputs s_1.

Therefore if the adversary is also given an encryption $(\overline{C}_2, \overline{R}_{2,1}) \leftarrow \mathsf{Enc}(\overline{P}_2, s_1)$, he can invoke $R_{1,2}(\overline{C}_1)$ and $R_{2,1}(\overline{C}_2)$ to recover s_1, s_2 respectively. As described in the previous section, from any longer cycle of size up to n one can compute a functionally working encryption of s_1 under key 2 i.e., a program that on input s_2 outputs s_1, that can be fed as well to R to recover the secret key.

IND-CPA Security of Our Scheme: The modified scheme is still IND-CPA secure: unless one knows an encryption of the secret key, it is not possible to exploit this R. The program Q, included in the public key, can only be used to output programs that output the secret key if one knows an encryption of the secret key, so it is useless. More formally, we prove the following:

Theorem 3. *If $\mathcal{O}$ is a VBB obfuscator for $\mathcal{P}$ according to Definition 2, then the modified scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *described in this section is IND-CPA secure.*

Proof. The first part of the theorem goes is the same as Theorem 2. We thus restart the proof at the point where we argue that there exist no simulator able to distinguish between oracles (P, Q, C_x, R_x) and (P, Q, C_y, R_y).

Note that the oracle Q on input m returns $\mathcal{O}(R_m)$. Assuming the obfuscator $\mathcal{O}$ to be VBB secure, this cannot help $\mathcal{S}$ to distinguish the two cases any more than having access to the oracle Q^* which takes as input two pairs (q, v) where q specifies the query type and v the value to be queried on: on input $(0, m)$, Q^* chooses a random string r_m, stores (r_m, m) and returns r_m to $\mathcal{S}$. One can think of r_m as a "handle" that can be used to access a newly created oracle R_m. On input a query of the form (q, C) with $q \neq 0$, the oracle Q^* returns s if a value $r_m = q$ was stored before and $C(m) = s$ or $\perp$ in every other case. So the existence of an adversary $\mathcal{A}$ that breaks the IND-CPA game with non negligible probability implies that the obfuscator is not secure or that there exist a $\mathcal{S}$ such that:

$$\left|\Pr[\mathcal{S}^{P,Q^*,C_x,R_x} = 1] - \Pr[\mathcal{S}^{P,Q^*,C_y,R_y} = 1]\right| > \mathrm{negl}(k) \tag{1}$$

We can now define $\perp$ to be an oracle that always returns $\perp$. Note that (P, C_x) can be substituted by $\perp$ and the $\mathcal{S}$ will notice only with negligible probability (i.e., if it guesses s) and therefore

$$\left|\Pr[\mathcal{S}^{\perp,Q^*,\perp,R_x} = 1 - \Pr[\mathcal{S}^{P,Q^*,C_x,R_x} = 1]\right| = \mathrm{negl}(k)$$

Define $Q^{\perp}$ to return random strings on queries of the from $(0, m)$ and and $\perp$ otherwise. Note that $\mathcal{S}$ can only distinguish those two oracles if it queries them on input $(0, m)$ and then (r_m, C) with $C(m) = s$ thus

$$\left|\Pr[\mathcal{S}^{\perp,Q^{\perp},\perp,R_x} = 1 - \Pr[\mathcal{S}^{\perp,Q^{*},\perp,R_y} = 1]\right| = \text{negl}(k)$$

In the final step we replace R_x with R_y. The only way that any simulator can distinguish between the two oracles is by querying the oracle on a program that outputs s on input x, y, and as the simulator knows x, y this is equivalent to guessing s. Repeating all the steps backwards with y instead of x we get a contradiction with (1).

□

Acknowledgements: The authors would like to thank Amit Sahai for the mantra "When you cannot solve a problem, try obfuscation" and Matthew Green for interesting discussions.

References

[ABBC10] Tolga Acar, Mira Belenkiy, Mihir Bellare, and David Cash. Cryptographic agility and its relation to circular encryption. In *EUROCRYPT*, pages 403–422, 2010.

[BG10] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In *CRYPTO*, pages 1–20, 2010.

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

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

[BGK11] Zvika Brakerski, Shafi Goldwasser, and Yael Tauman Kalai. Black-box circular-secure encryption beyond affine functions. In *TCC*, pages 201–218, 2011.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In *CRYPTO*, pages 108–125, 2008.

[BR13] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. Cryptology ePrint Archive, Report 2013/563, 2013. `http://eprint.iacr.org/`.

[BRS02] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *Selected Areas in Cryptography*, pages 62–75, 2002.

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

[CGH12] David Cash, Matthew Green, and Susan Hohenberger. New definitions and separations for circular security. In *Public Key Cryptography*, pages 540–557, 2012.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, pages 93–118, 2001.

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

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. Cryptology ePrint Archive, Report 2013/451, 2013. `http://eprint.iacr.org/`.

[GH10] Matthew Green and Susan Hohenberger. Cpa and cca-secure encryption systems that are not 2-circular secure. *IACR Cryptology ePrint Archive*, 2010:144, 2010.

[Hof13] Dennis Hofheinz. Circular chosen-ciphertext security with compact ciphertexts. In *EUROCRYPT*, pages 520–536, 2013.

[HRSV11] Susan Hohenberger, Guy N. Rothblum, Abhi Shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. *J. Cryptology*, 24(4):694–719, 2011.

[KRW13] Venkata Koppula, Kim Ramchen, and Brent Waters. Separations in circular security for arbitrary length key cycles. Cryptology ePrint Archive, Report 2013/683, 2013. `http://eprint.iacr.org/`.

[Wee05] Hoeteck Wee. On obfuscating point functions. In *STOC*, pages 523–532, 2005.