Obfuscation for Cryptographic Purposes

Dennis Hofheinz[1], John Malone-Lee[2], and Martijn Stam[3]

[1] CWI, Amsterdam, Dennis.Hofheinz@cwi.nl
[2] University of Bristol, malone@cs.bris.ac.uk
[3] EPFL, Lausanne, martijn.stam@epfl.ch

Abstract. An obfuscation $\mathcal{O}$ of a function F should satisfy two requirements: firstly, using $\mathcal{O}$ it should be possible to evaluate F; secondly, $\mathcal{O}$ should not reveal anything about F that cannot be learnt from oracle access to F. Several definitions for obfuscation exist. However, most of them are either too weak for or incompatible with cryptographic applications, or have been shown impossible to achieve, or both.

We give a new definition of obfuscation and argue for its reasonability and usefulness. In particular, we show that it is strong enough for cryptographic applications, yet we show that it has the potential for interesting positive results. We illustrate this with the following two results:

1. If the encryption algorithm of a secure secret-key encryption scheme can be obfuscated according to our definition, then the result is a secure public-key encryption scheme.
2. A uniformly random point function can be easily obfuscated according to our definition, by simply applying a one-way permutation. Previous obfuscators for point functions, under varying notions of security, are either probabilistic or in the random oracle model (but work for arbitrary distributions on the point function).

On the negative side, we show that

1. Following Hada [12] and Wee [25], any family of deterministic functions that can be obfuscated according to our definition must already be "approximately learnable." Thus, many deterministic functions cannot be obfuscated. However, a probabilistic functionality such as a probabilistic secret-key encryption scheme can potentially be obfuscated. In particular, this is possible for a public-key encryption scheme when viewed as a secret-key scheme.
2. There exists a secure probabilistic secret-key encryption scheme that cannot be obfuscated according to our definition. Thus, we cannot hope for a general-purpose cryptographic obfuscator for encryption schemes.

Keywords: obfuscation, point functions.

1 Introduction

The obfuscation of a function (or, more generally, a program) should provide nothing more than the possibility of evaluating that function. In particular, from an obfuscation of a function, one should not be able to learn more than one can learn from oracle access to that function.

History and Related Work. Practical yet informal approaches to code obfuscation were considered in [13, 16]. The first theoretical contributions were made by Hada [12] who gives a security definition for obfuscations and relates it to zero-knowledge proof systems.

In their seminal paper [1], Barak *et al.* define a hierarchy of obfuscation definitions, the weakest of which is predicate-based and the strongest of which is simulation-based. They show that there are languages that cannot be obfuscated, even under the weakest definition that they proposed. Specifically, they show that there are (contrived) sets of programs such that no single obfuscation algorithm can work for all of them (and output secure obfuscations of the given input program). Hence, Barak *et al.* rule out the possibility of *generic* obfuscation (and the proof argument they give also applies to our notion), yet they leave room for the possibility of obfuscators for *specific* families of programs.

Goldwasser and Tauman Kalai present obfuscation definitions which model several types of auxiliary information available to an adversary [11]. They show general impossibility results for these definitions using *filtered functions* (functions whose output is forced to zero if the input is not of a special form). They also show that secure (without auxiliary information) obfuscations of *point functions* (functions that are 0 everywhere except at one point, see Definition 4.1) are automatically secure with respect to independent auxiliary information.

Even before a precise definition of obfuscation was formulated, positive obfuscation results could be given implicitly and in a different context for a special class of functions. Namely, Canetti [3] and Canetti *et al.* [4] essentially obfuscate point functions. The construction from Canetti [3] works for (almost) arbitrary function distributions and hence requires a very strong computational assumption. On the other hand, one construction from Canetti *et al.* [4] requires only a standard computational assumption, but is also proven only a for uniform function distribution. Both of these constructions are probabilistic and technically very sophisticated.

Positive results for the predicate-based definition of Barak *et al.* [1] were demonstrated by Lynn *et al.* [17] who show how to employ a random oracle to efficiently obfuscate the control flow of programs, which includes point functions.

Subsequently Wee showed how to obfuscate point functions in the standard model [25] (still predicate-based). Yet he only does this under very strong computational assumptions and for a weakened definition of obfuscation. Wee also shows that, at least under one of the original obfuscation definitions of Barak *et al.* [1], point functions can only be obfuscated under strong computational assumptions.

Finally, a generalisation of one-way functions can be found in the work of Dodis and Smith [6] who show how to obfuscate a proximity function.

Our Work. We concentrate on a new definition that is a variant of the simulation-based definition of Barak *et al.* [1]. We deviate from the notion of [1] only in that we consider probabilistic functions and also pick the function to be obfuscated according to a distribution and hence demand only "good obfuscations on average." (This is similar to Canetti's "oracle hashing" definition from [3], or to the "approximate functionality with respect to a particular input distribution" variant [1].) However, we stress that the impossibility results from [1] also carry over to such a weakened notion in a meaningful sense. In fact, a similar notion is already informally considered in [1, Discussion after Theorem 4.5]. That means that also for our notion, there can be no general-purpose obfuscators. Yet our goal is to consider obfuscations for specific applications such as obfuscating secret-key encryption schemes. We stress that there *are* secret-key encryption schemes that are unobfuscatable (see Remark 4.1), so general-purpose obfuscators cannot exist; then again, there are also schemes that can be obfuscated (e.g., a public-key encryption scheme when viewed as secret-key). We are interested in *specific* obfuscations, not in general-purpose obfuscation.

Intuitively, our variation makes sense in many cryptographic applications where the function to be obfuscated is chosen at random by a trusted party. This could for instance model the situation where an issuer of smartcards selects a signing key and then hardwires it on a smartcard.

To show the usefulness of our notion, we demonstrate that by obfuscating the encryption algorithm of an IND-CPA secure symmetric encryption scheme, we obtain an IND-CPA secure asymmetric scheme. As a sidenote, we prove that, surprisingly, the analogous result does *not* hold for IND-CCA schemes. The latter is not a consequence of our relaxed definition, but is inherent in all existing definitions of obfuscation. We also prove similar results (both positive and negative) concerning the construction of signature schemes from MACs using obfuscation.

Although we are not yet able to give a concrete (interesting) example of a public-key encryption scheme or a signature scheme produced using our new definition of obfuscation, we provide evidence that the definition is satisfiable in other contexts. In particular, we show that, given a one-way permutation, it is possible to obfuscate a point function *deterministically* and in a quite straightforward way in the standard model. Previous obfuscations of point functions (under different definitions) were either probabilistic and quite involved or in the random oracle model; this owes to the fact that they were geared to work for *arbitrary* distributions on the point function to be obfuscated.

Our definition does not overcome the impossibility results of Barak *et al.* [1]. Actually, following Hada [12] and Wee [25], we remark that any family of deterministic functions must be approximately learnable to be obfuscatable. We prove that in particular, it is not possible to obfuscate pseudorandom functions under our definition.

2 Previous Definitions

Barak *et al.* [1] discuss the obfuscation of an arbitrary Turing machine or circuit. This leads to the same notation for the description of a circuit and the function it evaluates. Moreover, from an adversary's point of view, the security does not depend on the particular way some function is implemented prior to obfuscation.

Under our definitions, the implementation or representation of a function prior to obfuscation is relevant for the security of the obfuscation. To emphasize this, we will make an explicit distinction between keys of a function on the one hand, and keyed functions on the other.

Let $\mathcal{F} = \{\mathcal{F}_k\}_{k\in\mathbb{N}}$ be a class of probabilistic functions $\mathcal{F}_k = \{F_K\}_{K\in\mathcal{K}_k}$ where all $F_K \in \mathcal{F}_k$ have an identical domain X_k. We call K the key and we assume there is some probabilistic polynomial time (in k) algorithm F that, on input of $K \in \mathcal{K}_k$ and $x \in X_k$, samples from $F_K(x)$.

Formally, we regard an obfuscator as a combination of two algorithms: a key transformation algorithm $\mathcal{O} : \bigcup_k \mathcal{K}_k \to \bigcup_k \mathcal{K}'_k$ that takes a key $K \in \mathcal{K}_k$ and returns the obfuscated key $K' \in \mathcal{K}'_k$; and a probabilistic polynomial time algorithm G that, on input a key $K' \in \mathcal{K}'_k$ and $x \in X_k$, samples from $F_K(x)$. Depending on the context, we will not always make explicit mention of G.

We are now ready to rephrase Barak *et al.*'s definition of obfuscation in the setting that we consider. We only give a uniform model and, for the moment, concentrate on obfuscating deterministic functions. Note that our model is slightly more restrictive, as a result of which polynomial slowdown is automatically satisfied. (This refers to the slowdown between the function and its obfuscation.)

Definition 2.1 (Universal Obfuscation). *A PPT algorithm $\mathcal{O}$ is a universal obfuscator for a class $\mathcal{F}$ of* deterministic *functions if the following holds.*

- *Approximate Functionality: For all $k \in \mathbb{N}$, for all $K \in \mathcal{K}_k$ and all $x \in X_k$ it holds that $F_K(x) = G_{\mathcal{O}(K)}(x)$ with overwhelming probability over the choices of $\mathcal{O}$.*
- *Virtual Black-Box: Loosely speaking, given access to the obfuscated key $\mathcal{O}(K)$ of a function F_K, an adversary cannot learn anything about the original function F_K that it could not learn from oracle access to F_K. Formal definitions follow.*

Note that we call Definition 2.1 "universal" to indicate that—in contrast to our own upcoming refinement of this definition—security for each individual function F_K to be obfuscated is required.

Barak *et al.* [1] give several ways to formulate the notion of not learning anything, two of which we recall below.

Predicate-Based Obfuscation This is based on *computing a predicate.* In this case the task of an adversary given the obfuscation $\mathcal{O}(K)$ is to compute any boolean predicate on K. That is to say, for any adversary and any boolean predicate π, the probability that an adversary computes $\pi(K)$ given $\mathcal{O}(K)$ is no greater than the probability that a simulator, given only oracle access to F_K, computes $\pi(K)$. This notion is formally defined by a slightly simpler, but equivalent, notion.

Definition 2.2 (Predicate-Based Universal Black-Box). *A probabilistic algorithm $\mathcal{O}$ for a family $\mathcal{F}$ of functions satisfies the weak universal black-box* *property if for all PPT D, there exist a PPT S and a negligible function ν such that for all k and all $K \in \mathcal{K}_k$,*

$$\Big|\mathsf{Pr}\big[K' \leftarrow \mathcal{O}(K) \;:\; D(K') = 1\big] - \mathsf{Pr}\big[S^{F_K}(1^k) = 1\big]\Big| \leq \nu(k) \ .$$

Simulation-Based Obfuscation This is based on *computational indistinguishability*. Under this formulation one does not restrict the nature of what an adversary must compute: it says that for any adversary, given $\mathcal{O}(K)$, it is possible to simulate the output of the adversary given only oracle access to F_K. The outputs of the adversary and the simulator must be computationally indistinguishable. It is easy to see that in fact it is necessary and sufficient to simulate the output of the obfuscator (thus removing the need to quantify over all adversaries). This equivalence has also been observed by Wee [25] who gives the following formulation.

Definition 2.3 (Simulation-Based Universal Black-Box). *A probabilistic algorithm $\mathcal{O}$ for a class $\mathcal{F}$ of functions satisfies the strong universal black-box property if there exists a PPT S such that for all PPT D there exists a negligible function ν such that for all k and all $K \in \mathcal{K}_k$*

$$\Big|\mathsf{Pr}\big[K' \leftarrow \mathcal{O}(K) \;:\; D(K') = 1\big] - \mathsf{Pr}\big[\tilde{K}' \leftarrow S^{F_K}(1^k) \;:\; D(\tilde{K}') = 1\big]\Big| \leq \nu(k) \ .$$

3 Our Definition

Inspired by existing impossibility results, we endeavour to find a definition of cryptographic obfuscation that both allows meaningful applications such as transforming secret-key cryptosystems into public-key systems, and at the same time is satisfiable. Recall that previous work on obfuscation uses a universal quantifier for the functions $F_K \in \mathcal{F}_k$ to be obfuscated. In contrast, we will assume a key distribution on the keys K and hence on the functions F_K that have to be obfuscated. For simplicity we will assume a uniform distribution on the keys.

First, we will define and discuss a new notion of obfuscation that is a relaxation of the simulation-based definition of Section 2. In Section 4 we will examine some applications and limitations of our new definition. In the following, we will put emphasis on the virtual black-box property. (However, we include a short discussion of an adaptation of the approximate functionality requirement.)

3.1 The Definition

Simulation-based obfuscation refers to computational indistinguishability: given only oracle access one can produce something indistinguishable from the obfuscator's output. Note that the most straightforward analogue of Definition 2.1 with a randomized key distribution is not very meaningful. Since the distinguisher does not know the key K, a simulator can make up a different key and obfuscate it, so trivial obfuscation would be possible. To prevent this (and end up with a more sensible definition), we additionally give the distinguisher, similarly to the simulator, *oracle access* to the function.

Definition 3.1 (Simulation-Based Virtual Black-Box Property). *An obfuscation $\mathcal{O}$ for a class of functions $\mathcal{F}$ has the* simulation-based virtual black-box property *iff for all PPT distinguishers D there is a PPT simulator S such that the following quantity is negligible in k.*

$$\left|\mathsf{Pr}\left[K \leftarrow \mathcal{K}_k : D^{F_K}(1^k, \mathcal{O}(K)) = 1\right] - \mathsf{Pr}\left[K \leftarrow \mathcal{K}_k : D^{F_K}(1^k, S^{F_K}(1^k)) = 1\right]\right|.$$

3.2 On the Approximate Functionality Requirement

The natural analogue of the "approximate functionality" requirement from Definition 2.1 for the case of function *distributions* would be the following. For all keys K and inputs x, with overwhelming probability over the random choices of $\mathcal{O}$, the obfuscation $G_{\mathcal{O}(K)}(x)$ has *exactly* the same distribution as $F_K(x)$. This is a very strong requirement, and we can actually relax this a little.

Definition 3.1 *An obfuscator $\mathcal{O}$ satisfies the* statistical functionality requirement for $\mathcal{F}$ *iff there exists a negligible function ν such that for all k, the following holds:*

$$\sum_{K,K'} \mathsf{Pr}[K, K' : K' \leftarrow \mathcal{O}(K), K \leftarrow \mathcal{K}_k] \max_x (\sigma(G_{K'}(x), F_K(x))) \leq \nu(k).$$

Here, σ is used to denote the statistical distance.

For deterministic functions, the requirement reduces to the statement that, with overwhelming probability over the choice of the key generation and the obfuscator $\mathcal{O}$, F_K and $G_{\mathcal{O}(K)}$ should be the same functions. This is similar to the approximate functionality of the universal definition [1, Definition 4.3], with the caveat that we take our probability over F_K as well. We note that all the results to follow do not depend on the choice of functionality requirement.

3.3 Comparison to Previous Definitions

The Definitions of Barak et al. Definition 3.1 differs in several aspects from [1, Definition 2.1]. First, Definition 3.1 requires security w.r.t. a randomly chosen key from a given set, whereas [1, Definition 2.1] demands security for every key in that set. In that sense, Definition 3.1 is a relaxation of [1, Definition 2.1] (although this does not protect Definition 3.1 from impossibility results for general-purpose obfuscation; see below).

On the other hand, Definition 3.1 requires a multi-bit output from the simulator, whereas [1, Definition 2.1] restricts adversary and simulator to a one-bit output. As [1, Definition 2.1] demands security for all keys in a set, this one-bit output can be seen as an approximation of a predicate on the key. In fact, when directly relaxing [1, Definition 2.1] by randomizing the distribution on the key, approximating a predicate on the key leads to a more sensible definition than simply comparing the probabilities for 1-output of adversary and simulator. Such a predicate-based formulation, even with randomly chosen key and a distinguisher with oracle access, is incomparable to our definition (since the predicate could be not computable in polynomial time).

Perfect One-Way Hashing and Point Functions. We note that a distribution on the keys (or, on the function to obfuscate) was already considered in other definitions, e.g., in the security definition for perfect one-way hashing (that is actually an obfuscation of a point function) from [3]. In the case of [3], security could be achieved as long as the distribution on the functions was *well-spread*, which basically means that a brute-force search for the function has only negligible success. Our results from Section 4.3 (that also concern an obfuscation of a point function) are formulated with a *uniform* distribution on the key.

In contrast to the very sophisticated construction from [3], our construction is quite simple: an obfuscation of a point function P_x, is $\Pi(x)$ for a one-way permutation Π. However, there *can* be well-spread distributions (different from the uniform one) on the key for which our point function obfuscation becomes insecure. (Imagine a one-way permutation that leaks the upper half of the preimage, and a distribution that keeps the lower half of the preimage constant.) In other words, the price to pay for the simplicity of our construction is the dependency on a *uniform* distribution of the key.

Also, the construction from [3] is "semantically secure" in the sense that any predicate on the hashed value (i.e., the key of the point function to be obfuscated) is hidden. Our construction from Section 4.3 does not guarantee this; just like the one-way permutation that is employed, our construction only hides the key in its entirety. This may have the disadvantage that in some applications, this might not be sufficient, and in particular not a meaningful "idealization" of a point function. However, in other settings (such as a password query), this may be exactly the idealization one is interested in.

Other Similar Definitions. Technically, Definition 3.1 is quite similar to [12, Definition 10] (the latter definition which is also formulated with a distribution on the keys). Essentially, the only difference is that [12, Definition 10] equips the distinguisher with an extra copy of the obfuscation instead of oracle access to the function. As argued by Hada [12], this leads to a very strong definition (that is in particular strictly more restrictive than ours).

Finally, the definitions from Wee [25, Section 5.2] are technically similar to ours, in that they allow the adversary a multi-bit output. These definitions suffer from strong impossibility results (in particular, a function must be *exactly* learnable for obfuscation); this is partly due to the fact that these definitions demand security for *all* keys in a given set. In our case, a function must be *approximately* learnable for obfuscation, and this enables, e.g., the obfuscation of point functions (see Sections 4.3 and 4.4).

3.4 Specific vs. General-Purpose Obfuscation

Impossibility of General-Purpose Obfuscation. As indicated, also Definition 3.1 suffers from certain impossibility results. First, the argument from [1, Section 3] works also for the case of a randomized key distribution, and hence there *are* certain (albeit constructed) examples of unobfuscatable function families. There are even less constructed examples, as we will show in Remarks 4.1 and 4.2, and in Section 4.4. In other words: there can be no general-purpose obfuscation.[4]

Specific Obfuscators. What we advocate here is to consider *specific* obfuscators for *specific* function families. For example, we will show (in Section 4.1) that obfuscating the encryption algorithm of a secret-key encryption scheme yields a public-key encryption scheme, and that such obfuscations (in principle at least) exist. However, our example that such obfuscations exist assumes a public-key encryption scheme in the first place. Plugging this example into the secret-key→public-key transformation gives (nearly) the same public-key encryption scheme one started with. So the following question arises:

What is Gained? First, the secret-key→public-key transformation can be seen, similarly to [5], as a technical paradigm to realize public-key encryption in the first place. In that context, a formalization of obfuscation can provide an interface and a technical guideline of what to exactly achieve.

Second, the mentioned impossibility results does not exclude that a sensible formulation of *what* can be obfuscated exists. In other words, there may be a large and easily describable class of functions which *can* be obfuscated. Universal, general-purpose obfuscators for this class may exist and provide solutions for applications which correspond to functions inside this class.

4 Results Concerning the Simulation-Based Virtual Black-Box Property

In this section we discuss two applications of Definition 3.1: transforming secret-key encryption into public-key encryption and transforming MACs into signature schemes. Although we are not yet able to give an example of such a construction we provide evidence that our definition is satisfiable by demonstrating how to obfuscate a point function using a one-way permutation. Finally we present an impossibility result concerning Definition 3.1, thereby demonstrating that obfuscation definitions should be tailored to the context in which one wishes to use them.

4.1 Transforming Secret-Key Encryption

When the idea of public-key cryptography was first proposed by Diffie and Hellman [5], they suggested that one way to produce a public-key encryption scheme was to obfuscate a secret-key scheme. This application of obfuscation was also suggested by Barak *et al.* [1].

A secret-key encryption scheme SKE consists of the following three algorithms.

[4] It is actually worse: there are function families that cannot be obfuscated even with very specific, case-tailored obfuscators.

– A PPT *key generation algorithm* $\mathsf{SKE.KeyGen}$ that takes as input 1^k for $k \in \mathbf{Z}_{\geq 0}$. It outputs a key K.
– A polynomial time *encryption algorithm* $\mathsf{SKE.Enc}$ that takes as input 1^k for $k \in \mathbf{Z}_{\geq 0}$, a secret key K, and a message $m \in \{0,1\}^*$. It outputs a ciphertext c. Algorithm $\mathsf{SKE.Enc}$ may be probabilistic or deterministic.
– A polynomial time *decryption algorithm* $\mathsf{SKE.Dec}$ that takes as input 1^k for $k \in \mathbf{Z}_{\geq 0}$, a key K, and a ciphertext c. It outputs a message m.

Functionality of the scheme requires that for all keys, encryption followed by decryption under the same key is the identity function (slight relaxations of this statement are possible).

A secret-key cryptosystem is IND-CPA secure if no adversary with access to an encryption oracle can pick two messages, of equal length, such that it can distinguish (still having encryption-oracle access) between encryptions of the two. This is the notion called *find-then-guess* CPA (FTG-CPA) security by Bellare *et al.* [2].

A public-key cryptosystem consists of the same three algorithms, but with the difference that the key generation algorithm now outputs two keys: one private and one public. The public key is used for encryption, the private key for decryption. The scheme is IND-CPA secure if no adversary with access to the public key (and hence an encryption oracle) can pick two messages, of equal length, the encryptions of which it can distinguish.

A secret-key encryption scheme is turned into a public-key encryption scheme by releasing as the public key an obfuscation $\mathcal{O}(K)$ of $\mathsf{SKE.Enc}(1^k, K, \cdot)$, the private key encryption algorithm using key K.

Note that the correctness requirement of an obfuscation may not guarantee that the public-key scheme obtained in this way functions correctly in terms of decryption of encryption being the identity function. In fact, this is guaranteed only with overwhelming probability. However, we ignore this issue here as one can always demand perfect correctness from the obfuscation (which would result in a public-key encryption with perfect functionality), or one can weaken the functionality requirement for public-key encryption schemes.

Remark 4.1 (On the obfuscatability of secret-key encryption). In the following, we simply assume a secret-key encryption scheme with obfuscatable encryption algorithm. One may wonder how realistic that assumption is. First, there *are* unobfuscatable secret-key cryptosystems; any scheme that enjoys *ciphertext integrity* [14] in a "publicly verifiable way" cannot be obfuscated. That is, imagine that a keypair of a digital signature scheme is made part of the secret key, and any ciphertext is signed using the signing key, while the verification key is included in every ciphertext. Then by the functionality requirement, an obfuscation must be able to sign messages (i.e., ciphertexts) under this signing key (note that the "real" verification key can be obtained by oracle access to encryption, so the obfuscator cannot make up a different signing key). However, by unforgeability of the signature scheme, no simulator can do so with oracle access to encryption only.

But, on the other hand, *specific* secret-key encryption schemes *can* be obfuscated: imagine a public-key encryption scheme where the public key is part of every ciphertext.[5] The ability to encrypt arbitrary messages can then be acquired with one black-box query to the encryption oracle, hence if we view such a scheme as secret-key encryption, its encryption algorithm can be obfuscated.

So we find ourselves in a situation where we cannot hope for an *all-purpose* obfuscation (for secret-key encryption). In contrast, we hope for efficient obfuscations of *specific* schemes. We are unfortunately not able to give concrete examples here; instead, we simply assume such obfuscations and see how we could benefit:

Theorem 4.1 *If a secret-key encryption scheme SKE that is IND-CPA is turned into a public-key encryption scheme using the method above with an obfuscator satisfying Definition 3.1, then the resulting scheme is an IND-CPA secure public-key encryption scheme.*

Proof. For the sake of brevity, we will write $\mathsf{E}_K(\cdot)$ for $\mathsf{SKE.Enc}(1^k, K, \cdot)$ and O_{E_K} for an obfuscation thereof. Let a PPT adversary $A = (A_1, A_2)$ be an adversary of the public-key scheme whose advantage is

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND-CPA}} = \Big|&\Pr[K \leftarrow \mathsf{SKE.KeyGen}(1^k),\, O_{\mathsf{E}_K} \leftarrow \mathcal{O}(K),\\ &(m_0, m_1, h) \leftarrow A_1(O_{\mathsf{E}_K}),\, b \leftarrow \{0,1\} \,:\, A_2(h, O_{\mathsf{E}_K}(m_b)) = b] - \frac{1}{2}\Big|.\end{aligned}$$

(In the above we have split the adversary in two and use h to denote state information it might wish to relay.) We must show that this advantage is negligible. By approximation of obfuscation, we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND-CPA}} \approx \Big|&\Pr[K \leftarrow \mathsf{SKE.KeyGen}(1^k),\, O_{\mathsf{E}_K} \leftarrow \mathcal{O}(K),\\ &(m_0, m_1, h) \leftarrow A_1(O_{\mathsf{E}_K}),\, b \leftarrow \{0,1\} \,:\, A_2(h, \mathsf{E}_K(m_b)) = b] - \frac{1}{2}\Big|,\end{aligned}$$

where $X \approx Y$ denotes that $|X - Y|$ is negligible.

If we view A as a distinguisher against the obfuscation (that chooses b on its own and uses its oracle access to $E_K(\cdot)$ to obtain $E_K(m_b)$), then Definition 3.1 guarantees the following. There must be a simulator S that, given only oracle access to E_K, produces an output O'_{E_K} indistinguishable from O_{E_K} from A's point of view, yielding

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND-CPA}} \approx \Big|&\Pr[K \leftarrow \mathsf{SKE.KeyGen}(1^k),\, O'_{\mathsf{E}_K} \leftarrow S^{E_K},\\ &(m_0, m_1, h) \leftarrow A_1(O'_{\mathsf{E}_K}),\, b \leftarrow \{0,1\} \,:\, A_2(h, \mathsf{E}_K(m_b)) = b] - \frac{1}{2}\Big|.\end{aligned}$$

[5] This trick was suggested by a TCC referee.

Now consider the adversary (A_1', A_2) of the symmetric scheme SKE that runs S^{E_K} to obtain O'_{E_K} and then runs $A = (A_1, A_2)$, replacing O_{E_K} with $O'_{\mathsf{E}'_K}$.

From the above it follows that

$$\mathsf{Adv}^{\mathsf{IND-CPA}} \approx \Big| \mathsf{Pr}[K \leftarrow \mathsf{SKE.KeyGen}(1^k), \\ (m_0, m_1, h) \leftarrow {A_1'}^{\mathsf{E}_K}(1^k),\ b \leftarrow \{0,1\}\ :\ A_2(h, \mathsf{E}_K(m_b)) = b] - \frac{1}{2} \Big|,$$

and since the term on the right hand side is negligible by assumption, it follows that A's advantage against the public-key scheme is negligible as well. □

A stronger security requirement for secret-key and public-key encryption schemes is indistinguishability of ciphertexts under adaptive chosen-ciphertext attacks (IND-CCA, see [23]; for secret-key schemes, this is also called FTG-CCA in [2]). This notion is very similar to IND-CPA security, only an attacker (who tries to distinguish encryptions of two self-chosen plaintexts) is also given access to a decryption oracle. (Obviously, that access is limited in the sense that decryption of the ciphertext the adversary should distinguish is not allowed.)

It is natural to ask whether an IND-CCA secure secret-key scheme can be directly converted to an IND-CCA secure public-key scheme by obfuscating the encryption function. The next theorem shows that the answer is unfortunately negative:

Theorem 4.2 *Assuming that there is an IND-CCA secure secret-key encryption scheme* SKE *with obfuscatable encryption algorithm. Then, there is also another obfuscatable, IND-CCA secure secret-key encryption scheme* SKE' *and an obfuscator* $\mathcal{O}'$ *for* SKE' *such that, after obfuscating the encryption function* $\mathcal{O}'$*, the result is* not *an IND-CCA secure public-key encryption scheme.*

Proof. Assume an IND-CCA secure secret-key encryption scheme SKE that is obfuscatable in the sense of Definition 3.1. Modify SKE into a scheme SKE' as follows: the modified key generation outputs (K, r) for a key K as produced by $\mathsf{SKE.KeyGen}$ and a uniformly chosen random k-bit string r. A message m is encrypted under key (K, r) to (c, d), where $c \leftarrow \mathsf{SKE.Enc}(1^k, K, m)$ and d is the empty bitstring. A ciphertext (c, d) is decrypted under secret-key (K, r) to $m \leftarrow \mathsf{SKE.Dec}(1^k, K, c)$ if d is the empty bitstring, to K is $d = r$, and to $\perp$ otherwise.

The IND-CCA security of SKE' can be reduced to that of SKE: say that A' is an IND-CCA adversary on SKE'. A corresponding adversary A on SKE can internally simulate A' and only needs to translate oracle calls accordingly. Namely, A appends to each SKE-encryption an empty component d so as to make it an SKE'-encryption; decrpytion queries (c, d) are answered depending on d: if d is empty, the query is relayed to A's own decryption oracle, otherwise A answers the query on its own with $\perp$. (Note that this ensures that A never asks for decryption of the challenge ciphertext, since by assumption A' does not do so.) Since A' can have only negligible probability in guessing r, this provides A'

with a view at most negligibly away from its own SKE'-IND-CCA game. Hence A is successful iff A' is, and thus, SKE' is IND-CCA secure because SKE is.

Consider an obfuscator $\mathcal{O}$ for the encryption algorithm of SKE that satisfies Definition 3.1. Modify $\mathcal{O}$ into $\mathcal{O}'$, such that obfuscations produced by $\mathcal{O}'$ append an empty bitstring d to encryptions. Furthermore, make $\mathcal{O}'$ include r in the obfuscation. Since only the encryption algorithm is considered for obfuscation, $\mathcal{O}'$ still satisfies Definition 3.1. (Specifically, a simulator S' for $\mathcal{O}'$ can be obtained from a simulator S for $\mathcal{O}$ by simply appending a uniformly selected k-bit string r to the output of S.)

However, applying $\mathcal{O}'$ to $\mathsf{SKE.Enc}$ yields a public key encryption scheme in which r is part of the public key. Any query of the form (c, r) to the decryption oracle can be used by an IND-CCA attacker to obtain K and thus break the scheme. $\square$

Note that the precondition in Theorem 4.2—namely, an *obfuscatable* IND-CCA secure secret-key encryption scheme—is satisfiable by an IND-CCA *public-key* encryption scheme with a the argument from Remark 4.1.

Although, by Theorem 4.2, a direct construction of an IND-CCA secure public-key scheme is not possible using obfuscation, this does not mean that Definition 3.1 is not useful in this context: using Theorem 4.1 combined with a generic conversion from, say, IND-CPA to NM-CPA such as [22], one still obtains at least a non-malleable public-key scheme.

4.2 Transforming Message Authentication Codes

Another obvious application of obfuscation is to transform message authentication codes (in which a secret key is used for authenticating *and* verifying a message) into signature schemes (in which a secret key is used for signing, *i.e.*, authenticating, a message, but in which the verification key is public). Intuitively, this could be done by obfuscating the verification algorithm of the message authentication code. To begin with we will introduce the necessary concepts.

A message authentication code MAC consists of the three PPT algorithms $\mathsf{MAC.Key}$, $\mathsf{MAC.Sign}$ and $\mathsf{MAC.Verify}$, where

- $\mathsf{MAC.Key}(1^k)$ outputs a secret key K,
- $\mathsf{MAC.Sign}(1^k, K, m)$ signs a message $m \in \{0,1\}^*$ under key K and outputs a signature μ,
- $\mathsf{MAC.Verify}(1^k, K, m, \mu)$ verifies a signature μ to the message m.

As a functionality (or, correctness) requirement, one demands that under any possible key K, verifying a legitimately generated signature should always succeed. Again, relaxations are possible.

We demand for security that it should be hard for a PPT adversary to come up with a valid message/signature pair without knowing the secret key. Here, the adversary may request signatures of messages of its choice from a signing oracle. (Of course, signatures which are obtained through that oracle do not count as successful forgeries.) Since we explicitly allow that the signing algorithm is probabilistic, there may be multiple signatures for a single message. Therefore, we also equip the adversary with an oracle for verifying messages. Formally, a message authentication code MAC is *secure under adaptive chosen-message attacks*, or *MAC-CMA secure*, if and only if, for all PPT adversaries A, the following probability is negligible.

$$\mathsf{Pr}\big[K \leftarrow \mathsf{MAC.KeyGen}(1^k),\ (m,\mu) \leftarrow A^{\mathsf{MAC.Verify}(1^k,K,\cdot,\cdot),\mathsf{MAC.Sign}(1^k,K,\cdot)}(1^k)\ :\ \mathsf{MAC.Verify}(1^k,K,m,\mu)=1\big]$$

Analogously, a weaker notion of security, namely *security under verify-only attacks*, or *MAC-VOA*, can be derived by omitting the signing oracle from the above definition. Note that we have restricted here to PPT adversaries; actually, there are even schemes that achieve *unconditional security* [24].

The natural public-key analogue of message authentication codes are digital signature schemes. These are identical to message authentication codes, only the key generation algorithm outputs two keys: one public key that is used for verifying signatures, and a private key that is used for signing messages. Correctness is defined as for message authentication codes. The security notions SIG-CMA and SIG-VOA are defined exactly analogously to their message authentication code counterparts (only the adversary is given the public verification key in addition to 1^k).

So, in a digital signature scheme, verification is public, whereas in a message authentication code it requires a secret key. Thus, the verification algorithm of a message authentication code is a natural candidate for obfuscation. In fact, by obfuscating the verification algorithm of a message authentication code, we syntactically obtain a digital signature scheme. (As in the case of secret/public-key encryption, we ignore the *perfect* functionality requirement; here either the functionality requirement on the obfuscation must be perfect, or the functionality definition for digital signature schemes must be weakened.)

Technically, this means that the key generation $\mathsf{SIG.KeyGen}$ of the transformed scheme outputs a secret key K obtained from $\mathsf{MAC.KeyGen}$ along with an obfuscation of the verification function, $\mathcal{O}(\mathsf{MAC.Verify}(1^k,K,\cdot,\cdot))$, (with hard-coded secret key) as the public key. (Signing is unchanged, and verification simply means using the algorithm given by the public key.)

Remark 4.2 (On the obfuscatability of message authentication). We will simply assume a message authentication code with obfuscatable verification algorithm. Again, one may wonder how realistic that assumption is. First, there are certain (artificial) MACs whose verification algorithm cannot be obfuscated. (In particular, there can be no general-purpose MAC authenticator.) Since this is less straightforward to see than the existence of unobfuscatable secret-key encryption schemes, we now sketch such a MAC. (This is basically the general construction from [1, Section 3] adapted to the MAC interface.)

Let MAC be a MAC-VOA secure MAC. Define MAC' through

- $\mathsf{MAC}'.\mathsf{KeyGen}(1^k) := (K,\alpha,\beta)$ for $K \leftarrow \mathsf{MAC.KeyGen}(1^k)$ and uniformly chosen $\alpha,\beta \in \{0,1\}^k$.

- $\mathsf{MAC}'.\mathsf{Sign}(1^k, K', m) = (0, \mu)$ for $\mu \leftarrow \mathsf{MAC}.\mathsf{Sign}(1^k, K, m)$, where K' is parsed as (K, α, β).
- $\mathsf{MAC}'.\mathsf{Verify}(1^k, K', m, (0, \mu)) = \mathsf{MAC}.\mathsf{Verify}(1^k, K, \mu)$
- $\mathsf{MAC}'.\mathsf{Verify}(1^k, K', m, (1, \mu)) = 1$ iff $\mu(\alpha) = \beta$, where μ is interpreted as an algorithm description.[6]
- $\mathsf{MAC}'.\mathsf{Verify}(1^k, K', m, (2, \mu, i)) =$"the i-th bit of β", but only if $\mu = \alpha$ (otherwise, $\mathsf{MAC}'.\mathsf{Verify}$ returns 0).

First, it is easy to see that with oracle access to $\mathsf{MAC}'.\mathsf{Verify}$, no valid (in the sense of $\mathsf{MAC}'.\mathsf{Verify}$) "signatures" of the form $(1, \mu)$ or $(2, \mu)$ can be generated. Hence, MAC' inherits MAC's MAC-VOA security.

But now consider a distinguisher D who, on input an obfuscation O of $\mathsf{MAC}'.\mathsf{Verify}(1^k, K', \cdot, \cdot)$, returns $O(0^k, (1, O'))$, where the algorithm description O' is constructed from O such that

$$O'(x) = O(0^k, (2, x, 1))|| \dots ||O(0^k, (2, x, k)).$$

Then, functionality of an obfuscation dictates that $O'(\alpha) = \beta$ with overwhelming probability, and hence, $\Pr[D(O) = 1]$ must be overwhelming. On the other hand, no simulator can (with non-negligible probability, and from oracle access to $\mathsf{MAC}'.\mathsf{Verify}$ alone) produce a fake obfuscation $\tilde{O}$ that satisfies $\tilde{O}'(\alpha) = \beta$, so $\Pr\left[D(\tilde{O}) = 1\right]$ is negligible. Hence, MAC' cannot be obfuscated in the sense of Definition 3.1.

On the other hand, there also *are* obfuscatable MACs. Similarly to the encryption setting, any MAC-VOA secure digital signature scheme can be converted into a MAC-VOA secure MAC that is obfuscatable: simply declare the verification key part of the (secret) MAC key K. The obfuscation of the verification algorithm is simply the verification key. To achieve simulation of this obfuscation in the sense of Definition 3.1, we cannot use the trick from the encryption setting, where the public key was part of every encryption. In our setting, the verification algorithm outputs only one bit, and we must take care not to make a trivial signature forgery possible. However, a simulation of obfuscation is possible by simply outputting a *fresh* verification key randomly. No distinguisher can, with oracle access to the "right" verification routine, distinguish the "right" verification key from an independently generated one; to do so, it would need to forge a signature, which would contradict the MAC-VOA security of the digital signature scheme.

Analogously to the results in the previous section, we can make two statements about the security of the digital signature schemes that are obtained by obfuscating the verification algorithm of a message authentication code:

Theorem 4.3 *Say that a message authentication code* MAC *is MAC-VOA. If* MAC *is turned into a digital signature scheme by obfuscating using the method above and using an obfuscator satisfying Definition 3.1, then the resulting scheme is a SIG-VOA secure digital signature scheme.*

[6] Here and in the further analysis, we ignore complexity issues; techniques similar to those from [1] can and must be used to make $\mathsf{MAC}'.\mathsf{Verify}$ PPT.

Proof. The proof is very similar to the proof of Theorem 4.1 (in particular, the idea is to first use the functionality and then the simulatability of obfuscation), so we omit it. $\square$

Theorem 4.4 *Assuming that there is a MAC-CMA secure message authentication code* MAC *with obfuscatable verification algorithm. Then there is also a MAC-CMA secure message authentication code* MAC' *and an obfuscator (in the sense of Definition 3.1)* $\mathcal{O}'$ *for* MAC'*'s verification function, such that the result if* not *SIG-CMA secure as a digital signature scheme.*

Proof. The proof is analogous to the proof of Theorem 4.2. First, assume an obfuscatable MAC-CMA secure message authentication code MAC. Modify MAC into a code MAC' by including a uniformly selected $r \in \{0,1\}^k$ to the secret key during key generation. Signing and verification take places as before, except that if $m = r$ is to be signed, the signing algorithm appends K to the signature. This authentication code is still MAC-CMA, since an attacker has negligible probability of guessing r.

Any obfuscation $\mathcal{O}$ of the verification function can be modified into another one $\mathcal{O}'$ that includes the second part r of the secret key. If $\mathcal{O}$ satisfies Definition 3.1 for MAC, then so does $\mathcal{O}'$ for MAC', since a simulator that is to simulate an obfuscation of the verification function can simply choose r by itself. (It cannot be detected in doing so by only looking at the verification algorithm.)

However, applying $\mathcal{O}'$ to the verification algorithm of MAC' obviously leads to a digital signature scheme that is *not* SIG-CMA secure: an attacker gets r as part of the public key and simply needs to submit it to the signing oracle to obtain the signing key K. $\square$

4.3 Deterministic Obfuscation of Point Functions

In this section we prove a concrete feasibility result for Definition 3.1 by showing how to obfuscate a point function, as defined below.

Definition 4.1 (Point Functions). *For* $k \in \mathbf{Z}_{\geq 0}$ *and* $x \in \{0,1\}^k$*, the* point function $P_x : \{0,1\}^k \to \{0,1\}$ *is defined by* $P_x(y) = 1$ *if* $y = x$ *and* 0 *otherwise. Define* $\mathcal{P}_k = \{P_x :\ x \in \{0,1\}^k\}$.

We now show how to obfuscate point functions under Definition 3.1. Note that the requirement that the obfuscation has the same functionality as the original function follows directly from the construction.

Theorem 4.5 *Let* Π *be a one-way permutation on* $\{0,1\}^k$ *with respect to the uniform distribution. Then the obfuscation* $\mathcal{O}(x) = \Pi(x)$ *satisfies Definition 3.1 with respect to the uniform distribution on* $\mathcal{P}_k$ *and where the obfuscated function on input* y *and* $\mathcal{O}(x)$ *outputs 1 iff* $\Pi(y) = \mathcal{O}(x)$.

Proof. Consider the simulator S that outputs a uniformly sampled $y \in \{0,1\}^k$. We need to show that the difference from Definition 3.1 is negligible for any PPT distinguisher D. This is done by

$$\begin{aligned}
&\mathsf{Pr}\big[x \leftarrow \{0,1\}^k \,:\, D^{P_x}(1^k, \Pi(x)) = 1\big] \\
=\ &\mathsf{Pr}\big[y \leftarrow \{0,1\}^k \,:\, D^{P_y}(1^k, \Pi(y)) = 1\big] \\
\overset{(*)}{\approx}\ &\mathsf{Pr}\big[x, y \leftarrow \{0,1\}^k \,:\, D^{P_x}(1^k, \Pi(y)) = 1\big] \\
=\ &\mathsf{Pr}\big[x \leftarrow \{0,1\}^k \,:\, D^{P_x}(1^k, S(1^k)) = 1\big],
\end{aligned}$$

where $X \approx Y$ means that $|X - Y|$ is negligible in k. Here, $(*)$ can be shown by a reduction to the one-way property of Π. If there was a D for which $(*)$ does not hold, this D can be used to invert Π with non-negligible probability. A probabilistic Π-inverter D' then internally simulates D and works as follows. If D makes in any case at most, say, $p(k)$ oracle queries, D' chooses $i \in \{1, \dots, p(k)\}$ uniformly and answers all queries up to the i-th with 0 and outputs the i-th query as a guess for a preimage. □

Note 4.1. Very recently, [20] investigated to what extent point function obfuscations can be used to bootstrap other obfuscations. They did this under a definition of obfuscation in which adversaries are bounded *only* in their number of oracle queries, but not in the number of their computation steps. With respect to this definition, [20] shows that there are circuits which can be obfuscated with a random oracle, but not with just an oracle to a point function.

They also improve an upper bound on the concurrent self-composability of Wee's construction for point function obfuscation. Note that our construction, under our definition, is self-composable, which follows easily from the obfuscator being deterministic.

4.4 An Infeasibility Result

We conclude our work on Definition 3.1 by considering impossibility results on previous notions of obfuscation. The two notions that come closest to the new notion are simulation-based universal black box (Definition 2.3) and obfuscation with respect to independent auxiliary input [11, Definition 4].

Wee [25] shows that in the standard model obfuscation of deterministic functions with respect to Definition 2.3 is possible if and only if the functions are efficiently and exactly learnable (meaning that with a polynomial number of queries and effort one can construct a circuit that computes the function exactly). Since point functions are not efficiently and exactly learnable, it is clear from our positive result in the preceding section that Definition 3.1 is indeed a relaxation.

However, for a deterministic function one possible distinguisher simply samples random inputs and checks whether the obfuscated function (or simulated one) gives the same output as the real function (to which the distinguisher has oracle access). Consequently, the simulated function needs to correspond to the real function on all inputs, except for a small (and hard to find) fraction. Hence a function should be efficiently *approximately* learnable, that is, with a polynomial number of queries and effort one can construct a circuit that computes the function except on a small (and hard to find) fraction of the inputs. This in particular rules out the obfuscation of deterministic signature schemes, public-key decryption and pseudorandom functions.

To give a taste of a formal proof of the above, we give an explicit theorem and proof for the impossibility to obfuscate a pseudorandom function [8], as defined below.

Definition 4.2. *[8] A family of functions $\mathcal{F} = \{f_k\}$, $f_k : \{0,1\}^k \to \{0,1\}^k$ is* pseudorandom *if any probabilistic polynomial time algorithm D the advantage*

$$\Big|\big[f \leftarrow f_k \ : \ D^f(1^k) = 1\big] - \mathsf{Pr}\big[r \leftarrow r_k \ : \ D^r(1^k)\big]\Big|$$

is negligible in k where r_k is the set of all functions from $\{0,1\}^k$ to $\{0,1\}^k$.

Theorem 4.6 *It is impossible to obfuscate a pseudorandom function under Definition 3.1.*

Proof. Suppose for contradiction that there is an obfuscator $\mathcal{O}$ that satisfies Definition 3.1 when applied to a pseudorandom function family $\mathcal{F} = \{f_k\}$. For $f \leftarrow f_k$, consider the distinguisher D that, on input a supposed obfuscation g and with oracle access to f, chooses $x \in \{0,1\}^k$ and compares $f(x)$ with $g(x)$. By functionality of the obfuscation, in case $g = \mathcal{O}(f)$, we may assume that $f(x) = g(x)$ with overwhelming probability.

Now by assumption, there is a simulator S for this distinguisher that satisfies Definition 3.1. This S must thus be able to produce a function g with $f(x) = g(x)$, but it has only negligible probability of guessing x and thus, with overwhelming probability, does not query its f-oracle at x.

Thus S can be used to predict $f(x)$ with overwhelming probability without explicitly querying f. Hence S can be modified into a distinguisher that distinguishes f from a truly random function r as in Definition 4.2, which contradicts the pseudorandomness of $\mathcal{F}$. $\square$

To conclude, in addition to potential applications, the results of this section demonstrate that while it is satisfiable, Definition 3.1 is not appropriate for all application scenarios.

5 Conclusion

We have presented a simulation-based definition that, on the one hand, allows for obfuscating point functions, yet at the same time is strong enough for converting secret-key cryptography into public-key cryptography.

We would like to stress again that we do *not* rule out unobfuscatability results. In fact, we have shown certain scenarios in which obfuscation is not possible. On the other hand, our positive results (in particular the simplicity of our point function obfuscation) leave hope that obfuscations in interesting cryptographic scenarios are possible. We have given toy examples for the case of secret-key encryption or message authentication.

Acknowledgements

We are indebted to the Crypto 2006 and TCC 2007 referees who gave very valuable comments that helped to improve the paper. Specifically, the construction from Section 4.4 was suggested by one referee, and in particular one TCC referee had very constructive comments concerning presentation of our results. We also thank Alex Dent for motivating discussions.

References

1. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, 2001. Full version available at `http://eprint.iacr.org/2001/069/`.
2. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In 38^{th} *Annual Symposium on Foundations of Computer Science*, pages 394–403. IEEE Computer Science Press, 1997.
3. R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469. Springer-Verlag, 1997.
4. R. Canetti, D. Micciancio, and O. Reingold. Perfectly one-way probabilistic hash functions. In 30^{th} *ACM Symposium on Theory of Computing*, pages 131–140. ACM Press, 1998.
5. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
6. Y. Dodis and A. Smith. Correcting errors without leaking partial information. In 37^{th} *ACM Symposium on Theory of Computing*, pages 654–663. ACM Press, 2005.
7. R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In *Theory of Cryptography, TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 258–277. Springer-Verlag, 2004.
8. O. Goldreich, S. Goldwasser and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4), pages 210-217, 1986.
9. O. Goldreich and L. Levin. A hard-core predicate to any one-way function. In 21^{st} *ACM Symposium on Theory of Computing*, pages 25–32. ACM Press, 1989.
10. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
11. S. Goldwasser and Y. Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In 46^{th} *IEEE Symposium on Foundations of Computer Science*, pages 553–562. IEEE Computer Society, 2005.

12. S. Hada. Zero-knowledge and code obfuscation. In *Advances in Cryptology - ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 443–457. Springer-Verlag, 2000.
13. R. Jaeschke. Encrypting C source for distribution. *Journal of C Language Translation*, 2(1), 1990.
14. J. Katz and M. Yung. Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation. In *Fast Software Encryption - FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 284–299. Springer-Verlag, 2001.
15. P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.
16. C. Linn and S. Debray. Obfuscation of executable code to improve resistance to static disassembly. In 10^{th} *ACM Conference on Computer and Communications Security*, pages 290 – 299. ACM Press, 2003.
17. B. Lynn, M. Prabhakaran, and A. Sahai. Positive results and techniques for obfuscation. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 20–39. Springer-Verlag, 2004.
18. S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In *Theory of Cryptography, TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer-Verlag, 2004. Full version available at `http://eprint.iacr.org/2003/120/`.
19. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attack. In 22^{nd} *ACM Symposium on Theory of Computing*, pages 427–437. ACM Press, 1990.
20. Arvind Narayanan and Vitaly Shmatikov. On the Limits of Point Function Obfuscation. IACR ePrint Archive, May 2006. Online available at `http://eprint.iacr.org/2006/182.ps`.
21. National Institute of Standards and Technology. *Data Encryption Standard (DES)*, 1993. FIPS Publication 46-2.
22. R. Pass and a. shelat and V. Vaikuntanathan. Construction of a Non-Malleable Encryption Scheme From Any Semantically Secure One. In *Advances in Cryptology - CRYPTO '06*, volume 4116 of *Lecture Notes in Computer Science*. Springer-Verlag, 2006.
23. C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, 1992.
24. D. R. Stinson. *Cryptography: Theory and Practice*. CRC Press, 1995.
25. H. Wee. On obfuscating point functions. In 37^{th} *ACM Symposium on Theory of Computing*, pages 523–532. ACM Press, 2005.
26. A. C. Yao. Theory and applications of trapdoor functions (extended abstract). In 23^{rd} *Annual Symposium on Foundations of Computer Science*, pages 80–91. IEEE Computer Science Press, 1982.