Applicability of Indistinguishability Obfuscation

Seminar report for Research Seminar in Cryptography

Pille Pullonen

Supervisor: Dominique Unruh

December 14, 2014

1 Introduction

The intuitive meaning of obfuscation is taking a program and turning it into a program with the same functionality but somehow unintelligible structure. There have been practical approaches at obfuscating computer programs, but until 2001 [BGI+01] there was not much theory in this field. Especially, it was not well defined what is meant by obfuscation.

The main step of specifying the meaning of obfuscation would be to precisely define this unintelligibility. In the strongest sense we would like to specify that nothing is leaked about the program except the respective input and output behaviour. This has been formalised as virtual black-box (VBB) obfuscation and deemed impossible [BGI+01, BGI+12]. The next best currently known approach for defining obfuscation is indistinguishability obfuscation. The main benefit is that we know that indistinguishability obfuscation is achievable for all circuits [GGH+13]. On the other hand, indistinguishability obfuscation only guarantees that the two programs with the same functionality are indistinguishable when obfuscated.

The aim of this work is to introduce the topic of indistinguishability obfuscation by exploring the usefulness of this concept. The level on which the properties and details of different proposed schemes are described in this paper varies a lot. In some cases this paper also tries to give an overview of how indistinguishability obfuscation is used in the proposed applications. Especially, the aim is to outline the most common or more easily understandable approaches. However, in many cases this paper serves as a reference pointing to the ideas rather than discussing the nature of many quite technical constructions. In addition, many cryptographic primitives or ideas are not explicitly defined. The main hope is that every reader would find an area of their interest and see how obfuscation has been applied to advance this field.

In the following, Sec 2 defines what we mean by indistinguishability obfuscation and discusses what is provided by this concept. The second half of the paper in Sec 3 lists schemes where obfuscation has been used. The described results include both practical

constructions based on obfuscation as well as new theoretical advancements available thanks to the fact that indistinguishability obfuscation exists.

2 Indistinguishability Obfuscation

Strong version of indistinguishability that would be natural to consider is unachievable for general circuits. The definition of virtual black-box obfuscation as well as its impossibility were established by [BGI+01, BGI+12]. However, they also introduced the notion of indistinguishability obfuscation as a substitute definition to consider. Furthermore, it has been established that this notion is achievable [GGH+13] and is currently the best notion that we know for formalising obfuscation.

2.1 Definition

The obfuscation scheme should scramble the program so that it still computes the same functionality. However, for security we also want to define some properties about the program that the scrambling is able to hide. The definition of indistinguishability obfuscation $i\mathcal{O}$ [GGH+13] is given as follows.

Definition 1 (Indistinguishability obfuscator ($i\mathcal{O}$))**.** A uniform probabilistic polynomial time (PPT) machine $i\mathcal{O}$ is called an *indistinguishability obfuscator* for a circuit class $\{\mathcal{C}_\lambda\}$ if:

- *(Correctness)* For all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$ and all inputs x, we have that
$$\Pr[C'(x) = C(x) | C' \leftarrow i\mathcal{O}(\lambda, C)] = 1 \ .$$
- *(Security)* For any PPT distinguisher D, there exists a negligible function ε such that the for all security parameters $\lambda \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\lambda$, we have that if $C_0(x) = C_1(x)$ for all inputs x then
$$|\Pr[D(i\mathcal{O}(\lambda, C_0)) = 1] - \Pr[D(i\mathcal{O}(\lambda, C_1)) = 1]| \leq \varepsilon(\lambda) \ .$$

The first is the correctness condition, establishing that on all inputs the obfuscated version of the circuit computes the same result as the original circuit. The implication is that it always computes the same function.

The second part is the security definition of $i\mathcal{O}$. The idea is that for any pair of circuits that compute the same functionality, the respective obfuscated circuits should be indistinguishable.

2.2 Meaning of the definition

The main meaning of the $i\mathcal{O}$ definition is that no adversary can tell apart two obfuscated circuits that compute the same functionality and are of the same size. Moreover, it is

easy to see that such obfuscators exist as one can just define the obfuscation result as the first circuit of the given size and functionality [BGI+12]. This also gives a hint that learning the functionality of the obfuscated program or any other property that can be learned from any circuit of the same size and functionality does not break the security of obfuscation. However, distinguishing between the structures of the obfuscated programs is a security breach.

From the previous definitions we also know that the obfuscator is a function that maps the input circuit to a circuit of comparable size and the same functionality. For example, an obfuscation of a boolean function could be some normal form of that boolean expression.

Note that this definition does not provide any guarantees about obfuscating two circuits with different functionality. However, [BGI+12] discusses also this case and concludes that if it is possible to distinguish these circuits then it must also be possible to find inputs on which they differ. In addition, they also argue that $i\mathcal{O}$ is as good as any other obfuscator in such cases. However, this topic has emerged as a different research direction known as differing-inputs obfuscation [ABG+13].

2.3 Usefulness of $i\mathcal{O}$

It is not immediately clear where or whether this definition is useful or what is hidden about the circuit when it is obfuscated. For example, a special case where we can not derive anything from this definition is when there exists only one circuit for a specified functionality in the circuit class $\mathcal{C}_\lambda$. In such case we could not find a circuit pair with the same functionality and an $i\mathcal{O}$ would not have to do anything, for example. Even in cases where several circuits for the same functionality are allowed, there might be properties leaked by each of such circuits, that are therefore also leaked by the obfuscation. Hence, the fact that two circuits of the same functionality can not be distinguished after obfuscation does not give any explicit guarantees regarding what besides the input and output behaviour of these functions is derivable from the obfuscation.

The notion of best-possible obfuscation [GR07] is closely related to $i\mathcal{O}$. Especially, each best-possible obfuscator is $i\mathcal{O}$ and these definitions coincide for efficient obfuscators, but do not coincide for inefficient ones unless the polynomial hierarchy collapses. The definition of the best possible obfuscator specifies that anything that can be learned from an obfuscated program can be learned from any other program of the same functionality and similar size. The term *best-possible* results from this definition as such obfuscator maps each input circuit to a best possible circuit of the same functionality in terms of information leakage as it leaks less information than any other possible circuit of this functionality. Ultimately, the formal definitions of best-possible and indistinguishability obfuscation are different as the former is based in simulatability of the obfuscation. Note that neither of the definitions guarantees what exactly is hidden by the obfuscator, but the best-possible obfuscator definition specifies that whatever is hidden, is the best possible we can do. Especially, as $i\mathcal{O}$ is currently the best that we know how to construct, thanks to the definition of best-possible obfuscation it might be that we are in the future

able to come up with more intuitively usable definitions for obfuscation, but can also use the results obtained for $i\mathcal{O}$ in connection to this new definition.

Note that $i\mathcal{O}$ alone can not be used to provide interesting cryptographic constructions [SW14, GGHR14]. Especially, as $i\mathcal{O}$ would also exist if $P = NP$, but many cryptographic primitives would not. A trivial obfuscator is such that takes the lexicographically first circuit of the required functionality and it can be computed if $P = NP$ as in this case the polynomial hierarchy collapses. For that reason the following cryptographic constructions also use additional assumptions, such as the existence of one-way functions. On the other hand, if interesting cryptographic constructions can be built solely from $i\mathcal{O}$ then also $P \neq NP$ is implied.

Currently, it is interesting to study $i\mathcal{O}$ as this is the best definition that we know for obfuscation that can be achieved. However, actually using it can be tricky as by definition, we can only give security guarantees for obfuscations of circuits that compute exactly the same functionality.

2.4 Punctured Programs

One general approach for using $i\mathcal{O}$ has been proposed by [SW14] that considers punctured programs as a basis for applying $i\mathcal{O}$ in various settings. The idea is to remove some key element of the obfuscated program that does not change the functionality. In addition, the key element on which the program is punctured should be secret for the adversary. If the punctured value is at some location never accessed by the functionality then the two programs are functionally equivalent and therefore $i\mathcal{O}$ makes their obfuscated versions indistinguishable.

The central tool of their work are the punctured pseudo-random functions (PRF) that are defined on all strings of fixed length, except for some polynomial size group of these strings. These PRFs are known to exist if one-way functions exist. Commonly, some value defined by the security game of the required scheme is punctured out of the PRF, however, it is made sure that it is unlikely that the PRF would be queried on that value anyway. In this case, the punctured and un-punctured programs are likely to have the same input-output behaviour. For example, the input to PRF might be an output of a pseudo-random generator that has a sparse image. Then, when choosing a random puncturing point, it is likely that the punctured point is not within that image. Hence, the real program can be changed to the punctured program. However, from the purpose of the rest of the security game the point that is queried is exactly the punctured point. In this case the output of the PRF is chosen uniformly at random from the range space of the PRF. This allows to replace the outcome of the PRF in this game with a uniformly random value. As a concrete example, see the description of the public key encryption scheme in Sec 3.2. Especially, this relies on the security definition that given access to the punctured version of the PRF, the values of the original PRF in the punctured points are indistinguishable from randomly chosen values.

They use this approach to build several cryptographic constructions. For example, short signatures, public key encryption, CCA-secure encryption, perfect non-interactive

zero-knowledge arguments for NP, injective trapdoor functions, oblivious transfer and deniable encryption. We comment on some of those in the following section. Note however, that they also require the assumptions for existence of one-way functions. As a side-note, they expect that most cryptographic primitives can be built from $i\mathcal{O}$ and one-way functions.

2.5 Extensions of Indistinguishability Obfuscation

2.5.1 Differing Inputs Obfuscation

A notion connected to $i\mathcal{O}$ is differing inputs obfuscation proposed in [BGI+01]. This tries to establish similar setting as the indistinguishability obfuscation, but not requiring that the circuits compute exactly the same function. Especially, the obfuscations should be distinguishable only if an input, where the outputs of the circuits differ, is known. Correspondingly, special setting for differing inputs obfuscation [ABG+13] or extractability obfuscation [BCP14] have been studied. The idea is that if it is hard to find an input where two circuits differ then it should be impossible to distinguish the obfuscations. This is a direct strengthening of $i\mathcal{O}$. However, it has been shown that if a specific obfuscation exists for a signature scheme, then differing inputs obfuscation is impossible [GGHW14]. Although neither of the constructions is known, however, it seems likely that there does not exist a differing inputs obfuscation.

2.5.2 Probabilistic Indistinguishability Obfuscation

The notion of $i\mathcal{O}$ has also been extended to probabilistic circuits, resulting in probabilistic indistinguishability obfuscation [CLTV14]. The idea there is that all circuits where the output distributions are computationally indistinguishable should be indistinguishable when obfuscated. The idea is that for $i\mathcal{O}$ all circuits are deterministic, but part of the input can serve as a randomness for algorithms that require it. However, in case of probabilistic circuits it is possible to hide the randomness used by the circuit from the party that provides the input. This might be useful for example to provide obfuscated re-encryption circuits to encrypt some already encrypted message with a fresh randomness that no party knows. Note however, that it is allowed that the output distributions of the function and the obfuscated function are also only computationally indistinguishable and not necessarily the same. However, the result of the obfuscation is in fact a deterministic circuit where the obfuscation procedure fixes the randomness. Therefore the distinguisher is not allowed to query the same message several times. [CLTV14] provides both an instantiation of such obfuscation as well as applications to FHE and deterministic $i\mathcal{O}$. Their construction for probabilistic indisdinguishability obfuscation is built from deterministic $i\mathcal{O}$.

3 Indistinguishability Obfuscation as a building block

This section lists numerous applications of $i\mathcal{O}$ in different areas of cryptography. These include some new constructions for known primitives as well as new theoretical results. For example, $i\mathcal{O}$ can be used to build deniable or functional encryption. On the other hand, it can be used to prove new results about applicability of the random oracle model or circular security.

3.1 Software Obfuscation

Although the obfuscation ideas used in practice for obfuscating software do not follow the theory considered in this document, there are some proposals also how we might apply indistinguishability obfuscation also in case of software obfuscation. However, note that these are not well formalised for practical use as current proposed schemes operate mostly on circuits or Turing machines and it might not be straightforward to use them for real life software.

The first application proposed for $i\mathcal{O}$ was removing software watermarks [BGI+12]. Assuming that the watermarked software has to have the same behaviour as non-watermarked software, we know that the obfuscations of these two would be indistinguishable. Hence, obfuscation effectively removes the watermark as we get the same functionality but no way to test if the initial program was watermarked.

Together with the first construction of $i\mathcal{O}$ another software related application of $i\mathcal{O}$ was proposed by [GGH+13]. They consider restricted-use software. For example, if someone wants to create a demo version of their software that has limited functionality compared to the full version then they could just turn off some features at the level of the interface, but not by deleting the code. The main risk is that someone reverse engineers the closed version to obtain access to the full functionality. However, the limited interface version has exactly the same functionality as the version that just contains the code necessary for the demo. Hence, the solution against reverse engineering would be to use indistinguishability obfuscation. We know that in this case the obfuscation of the limited interface version would be the same as the actual program with only the limited functionality. Hence, it would be impossible to reverse engineer the full functionality as this is hidden by the indistinguishability obfuscation.

3.2 Public key encryption

To further illustrate the popular use of the punctured PRF approach to using $i\mathcal{O}$ we consider the example of a public key encryption from [SW14]. Note that some details have been omitted to simplify the discussion, in case of any questions see [SW14]. Assume that we have a puncturable PRF and a PRG. The encryption is defined as

$$\mathsf{Enc}(m, r) = (PRG(r), PRF(K, PRG(r)) \oplus m)$$

for a randomness r, secret key K and a message m. More specifically, the public key is an obfuscation of this functionality. To see the usage of the puncturing we have to

consider the IND-CPA security of this scheme. The initial game is defined as follows.

1. A random value r, $t = PRG(r)$ and a key K are chosen.
2. Public key is computed and given to the adversary.
3. The adversary gives two messages m_0 and m_1.
4. The challenge $\mathsf{Enc}(m_b, r)$ for a uniformly chosen b is forwarded to the adversary.

The first modification of this game is such that t is chosen uniformly instead of r and is used instead of $PRG(r)$ when computing the challenge. This is allowed based on the security of the PRG. Secondly, the encryption function used for the public key is modified so that the used PRF is punctured at t. It is unlikely that t is in the image of the PRG, hence the functionality of the encryption is still the same and the security is preserved by the security of $i\mathcal{O}$ as the public key is the obfuscation of the encryption function. Finally, the output of the initial PRF t is indistinguishable from random element for an adversary who only has access to the punctured PRF, hence, the final challenge $\mathsf{Enc}(m_b, r)$ can be replaced by $t, r*$ for a uniformly random r^*. The security of this step is ensured by the properties of the punctured PRF as it can not give information about the original PRF in the punctured points. However, in total we have obtained that the output of the encryption is indistinguishable from two random values and hence the scheme is IND-CPA secure.

Note that the decryption of this scheme can be computed efficiently as given the ciphertext $\mathsf{Enc}(m, r) = (c_1, c_2)$ we have $m = PRF(K, c_1) \oplus c_2$. Hence, it is a functional public-key encryption scheme with a secret key K.

3.3 Multi-linear Maps

A multi-linear map is a function with multiple inputs that is linear with respect to each separate input.

3.3.1 Self-bilinear Map

Self-bilinear maps with auxiliary information are constructed based on $i\mathcal{O}$ [YYHK14], this construction is used to obtain multi-linear maps where multilinearity level is not bounded during setup. In addition, the size of the group elements is independent of the multilinearity level. On the downside, additional information is required to evaluate this multi-linear map.

They consider groups of unknown composite order, especially the signed quadratic residues $\mathbb{QR}_N$ where N is a Blum integer. The idea is to define the mapping $e(g^x, g^y) = g^{2xy}$. Especially, they need to ensure that computational Diffie-Hellman assumption holds even in the presence of this mapping. However, computing this mapping itself is also difficult and requires some information τ_y for each g^y to compute $e(\cdot, g^y)$. Especially, τ_y could be a circuit that computes the $2y$'th power of its input, but such circuit can

leak the input y. This can be overcome by setting $t_y = 2y \pm ord(\mathbb{QR}_N)$ where the sign depends on the value of y. However, the final problem is that the order is unknown and therefore t_y can not be computed efficiently. To that end, $i\mathcal{O}$ is used to actually obfuscate the circuit that computes the $2y$'th power, but is indistinguishable from an obfuscation of the circuit that computes t_y'th power. Hence, for each particular computation of $e(g^x, g^y)$ the obfuscated circuit to compute $2y$'th power is needed.

This scheme can be used for non-interactive multi-party key exchange, distributed broadcast encryption, attribute based encryption, and constructing somewhat homomorphic encryption. Their main contribution is to show that the map with auxiliary information is sufficient for these applications of multi-linear maps.

3.3.2 Applications of Multi-linear Maps with $i\mathcal{O}$

As seen from [YYHK14], multi-linear maps are useful for non-interactive key exchange, broadcast encryption and somewhat homomorphic encryption. There are also other schemes that can be obtained from multi-linear maps. Boneh and Zhandry propose some of those based on $i\mathcal{O}$ in [BZ14] without directly using multi-linear maps. However, note that current $i\mathcal{O}$ schemes are based on multi-linear maps. On the other hand, it is not known how to construct the multi-linear maps only from $i\mathcal{O}$. The constructions in [BZ14] use the punctured PRF idea from [SW14].

Firstly, they propose multi-party non-interactive key exchange (NIKE) based on $i\mathcal{O}$ and PRG. In NIKE each party posts a public message for which it knows some secret that allows it to combine the rest of the public messages to form a key. In [BZ14], NIKE is obtained so that each party picks a random seed and evaluates the PRG on that seed to obtain its public message. In addition, there exists an obfuscated program of a PRF with a fixed key that no party knows. The shared key is computed by this PRF on all public inputs, but only if the secret information to one of the public values is also provided. The latter ensures that only the fixed parties in the computation can obtain the secret key. This is secure with respect to static passive adversaries and requires no trusted setup.

In addition, [BZ14] proposes schemes for broadcast encryption and recipient private broadcast encryption. Broadcast encryption allows a party to broadcast a message that only some subset of the parties can decode. Their basic construction is based on the NIKE scheme where only the desired subset is able to obtain the necessary key. In the recipient private keys no-one should know which of the other parties can decrypt the message. Their idea is to encrypt the recipient list in the broadcast header and to publish an obfuscated program that at first decrypts this list and only decrypts the message if the party can prove that it is in this list. As an application of broadcast encryption they also propose new schemes for traitor tracing. Traitor tracing allows to search for pirate decryption boxes.

3.4 Functional Encryption

Together with the first scheme for $i\mathcal{O}$ the application of functional encryption was also introduced [GGH+13] and has been a popular topic of $i\mathcal{O}$ hereafter [GJKS13, GGG+14, Wat14]. In functional encryption, the ciphertext encrypts input x, and there can be several secret keys SK_y for bitstrings y so that decrypting with a specific key SK_y results in $F(x, y)$ where we let $f_y = F(\cdot, y)$. Functional encryption has been an interesting topic since the start of attribute-based encryption, however, the term appeared in [SW08] and was formalised in [BSW11, O'N10].

Clearly different schemes provide different computation capabilities and $i\mathcal{O}$ based constructions have developed many new cases. Especially, before [GGH+13] it was unclear if functional encryption schemes exist for all polynomial size circuits. Garg et. al [GGH+13] established this by proposing first such scheme.

The security of functional encryption is indistinguishability based, especially requiring that for two adversary chosen messages m_0 and m_1 and each f_y chosen by the adversary we have $f_y(m_0) = f_y(m_1)$ and that the adversary can not distinguish the two encryptions. Note that besides obtaining the challenge encryption, the adversary can also obtain the secret keys SK_y for each f_y that satisfies the condition above. Note that thanks to the fact that the adversary gains the power to decrypt some functionalities this definition ensures that nothing except the function output can be learned using decryption. Otherwise the adversary might be able to combine a functionality that has an equal output for both chosen messages, but leaks some other information that allows him to distinguish the encryptions.

They also propose a IND-CCA secure bit-encryption scheme and public key encapsulation mechanism.

3.4.1 Construction

In principle, the functional encryption can be built from obfuscation using any public key encryption scheme. In such case the encryption would be as defined in the common public key encryption scheme. However, the decryption with key SK_y would be an obfuscation of a circuit that at first decrypts the message and then uses f_y to compute the output. This would work for black-box obfuscation, however, it is not clear that this is secure for indistinguishability obfuscation. Namely, it is not clear what the obfuscation hides about SK_y. For example, it does not guarantee that nothing besides the function output can be learned from the encryption. hence, it requires additional tricks to base this construction on $i\mathcal{O}$.

The initial construction in [GGH+13] however builds from this basic intuitive idea. They define a scheme with two keypairs of the initial public key encryption scheme. In such a scheme an encryption of x is a pair of encryptions of the original scheme using both public keys. The decryption uses an obfuscated circuit to check if the both ciphertexts indeed encrypt the same value and evaluate the decryption and function computation on one of the ciphertexts. However, for this to succeed the encryption must also contain a non-interactive zero-knowledge proof about the two encryptions

corresponding to the same message x. In this case the equality of the messages can be verified in the decryption by verifying the proof. The main reason for having two keys is that during the proof of security the obfuscated programs corresponding to the two secret keys are indistinguishable and therefore can be changed with each other. This as well as many of the subsequent schemes achieves selective security where the adversary has to choose the two messages m_1 and m_0 before seeing the parameters of the scheme. Their construction can be enhanced to full security.

3.4.2 Enhanced constructions

Indistinguishability obfuscaton based constructions have also been proposed for randomised functional encryption [GJKS13]. The main question that arises is the source of the randomness as it can not be specified alone by encryption or by decryption because in this case it would be susceptible to either corrupted party. Even simple combinations of both types of randomness sources may not be secure. Differently from the previous case they also consider simulation based security definition. It continues with the same idea that two keypairs are constructed, however the encryption has to produce a non-interactive witness indistinguishable proof about the equality of the two encrypted messages or the knowledge of a trapdoor. The randomness is computed by the decryption circuit using a specific pseudo random function.

Another branch regarding functional encryption has been multi-input functional encryption [GGG+14] where the circuit defined by the secret key SK_f can be a n-ary function. Note that [GGG+14] is a combination of [GKL+13, GGJS13]. In multi-input function encryption, instead of one ciphertext, the decryption is computed for a tuple of ciphertexts, in addition, different encryption key is allowed for each of the ciphertexts. Their general construction is similar to that of [GJKS13] still using two key pairs and a non-interactive witness-indistinguishable proof. However these initial keys are used to create different public keys for each input location. Besides the general construction they also consider other settings omitted from this summary. As a theoretically interesting result they also show that existence of a multi-input functional encryption scheme that achieves indistinguishability based definition implies an existence of a $i\mathcal{O}$.

Direct adaptive security where the adversary learns the parameters of the encryption scheme and can query some decryption keys before selecting the two messages is considered in [Wat14]. Their construction is based on puncturable pseudo random functions and only resembles the previous constructions in the fact that the secret key is an obfuscated program.

Some theoretical analysis of functional encryption complexity, especially that $i\mathcal{O}$ can be used to obtain communication efficient function evaluation can be found in [HW14].

3.4.3 Fully Homomorphic Encryption

Multi-input, especially two input functional encryption scheme can be used to construct a fully homomorphic encryption scheme [ABF+13]. Their idea is that the functionality supported by the FE scheme is a NAND gate and an encryption and all the functionalities

that we want to compute homomorphically can be viewed as circuits of these gates. The ability to compute encryption is required as the FE schemes decrypt the result and then compute the desired functionality. On the other hand, the homomorphic properties of FHE schemes allow to compute encryptions of function results from the encryptions of the arguments. Therefore, we need to define the function for FE so that it at first computes the function and then encrypts the result. However, additional care is required to ensure that the encryption computed as part of FE is properly randomised. To that end, randomised functional encryption, that can be built from regular FE, is introduced.

3.5 Zero-knowledge proofs

Zero-knowledge proofs to prove the correctness of inputs are an essential tool in many uses of $i\mathcal{O}$. However, we can also use $i\mathcal{O}$ to build those proofs.

One of the applications of the punctured programming approach of [SW14] is to obtain perfect non-interactive zero-knowledge (NIZK). They give a proof system for proofs of NP statements up to a fixed size based on two obfuscated programs. The first is an obfuscation of a proving algorithm that works based on the problem instance and the witness and outputs a signature for them. The second algorithm is the verification that verifies the signature.

Differing input obfuscation can also lead to four message concurrent zero-knowledge for NP language [PPS13].

3.6 Secure multi-party computation

In secure multi-party computation (MPC) the idea is that a set of participants needs to compute some function but does not want to reveal their inputs to this function. For example, the functional encryption can be used for this purpose when there are two parties and the function is fixed in advance. Some discussion regarding the complexity of secure function evaluation with FE can also be found in [HW14]. The rest of this section lists some usages of $i\mathcal{O}$ in secure multi-party computation.

3.6.1 Round complexity

The round complexity is one of the general measures of complexity of MPC. Using $i\mathcal{O}$ it is possible to construct the first two broadcast round general MPC protocol [GGHR14]. Note that we can not do better than two rounds for a general functionality [GP14]. These two-round protocols are universally composable (UC) secure against static malicious adversary and achieve fairness in case of an honest majority. Besides reducing the round complexity they also achieve low communication complexity for the rounds. They achieve such protocols using $i\mathcal{O}$ to transform each MPC protocol to a two-round protocol in a CRS model. At first the parties commit to their inputs and randomnesses and in the second round they provide an obfuscation of their code that can be evaluated by each party separately to compute all the messages otherwise computed in the protocol.

Besides $i\mathcal{O}$ CCA-secure public key encryption and non-interactive zero-knowledge are the main tools of the construction.

In more detail [GGHR14] uses CCA-secure cryptosystem to commit to the inputs. Secondly, any MPC protocol can be taken as an underlying construction. Each party creates obfuscated programs for computing each of its rounds and distributes these to other parties, whereas its inputs are hard-coded to the obfuscation. As a security measure the obfuscations have to be such that they can only be evaluated on the inputs and randomnesses that the other parties initially committed to. For this purpose the NIZK proofs are required to ensure the correctness of the inputs. Therefore, the obfuscations have to be able to also provide proofs that their outputs are consistent. For the security of $i\mathcal{O}$ it can be shown that either the real or simulated messages can be computed by the obfuscated programs. An additional assumption of multikey fully homomorphic encryption is required to reduce the communication complexity.

The same result has also been obtained in the adaptive security setting with active adversary [GP14]. They show that $i\mathcal{O}$ and common reference string are sufficient to obtain a two-round UC-secure protocol with broadcast messages for any functionality. The constructions for the adaptively secure scheme differ significantly from the static security scheme. As a main goal, the honest parties in the protocol should not generate any obfuscations. Especially, their core idea is that the common reference string could contain an obfuscation of the trusted third party. For adaptive security the scheme requires adaptively secure commitments and deniable encryption. Further tricks are needed to base this on $i\mathcal{O}$ rather than black-box obfuscation.

Simultaneously, a two-round adaptively passively secure two-party protocol was proposed [CGP14] following the punctured programming idea of [SW14]. Their construction is based on garbled circuits and the initial idea is to use adaptively secure OT and that the garbler's algorithm is obfuscated. Another constant round adaptively secure MPC protocol from garbled circuits based on the punctured PRF idea and explainability of algorithms [SW14] was proposed by [DSKR14].

3.6.2 Garbled Circuits

The garbled circuits based MPC has also been developed using $i\mathcal{O}$ to obtain succinct garbling schemes where the size and generation time is logarithmic in the running-time and polynomial in the size of the original program [LP14]. In their case the garbling is an obfuscated program. There are succinct obfuscation schemes proposed for Turing machines using differing-inputs obfuscation that could be used directly, however their construction uses regular $i\mathcal{O}$ schemes for circuits. Still, they only consider bounded-space computation where the size of the garbled circuits depends on the space complexity of the computation but not on the running-time.

Assuming the existence of $i\mathcal{O}$ and one-way functions then there exists succinct garbling schemes for all polynomial-time programs with polynomial space complexity. At first they obtain a non-succinct garbling that consist of several building blocks that can be generated independently and then use $i\mathcal{O}$ to compress the garbling. At the evaluation

time, the underlying garbled program gets decompressed in the obfuscated computation. The idea of the compression is to return an obfuscation of a randomized program that generates the garbled circuit.

Different applications of these schemes include succinct randomized encodings, functional encryption, garbled scheme based MPC, $i\mathcal{O}$ for RAM.

3.6.3 Secret Sharing

Secret sharing enables a set of parties to distribute a secret so that any qualified subset of the participants can restore the secret, but no unqualified set can. The idea with computational secret sharing is that dealing and restoring as well as the size of the shares should be efficient. Especially, the idea is to consider more general access structures than those of the traditional Shamir or additive secret sharing schemes. Using $i\mathcal{O}$ it is possible to propose a computational secret sharing scheme for any monotone NP functionality based access structure if we also have witness encryption and one-way functions [KNY14]. For a qualified set there exists a witness that can be used to obtain the secret. This result follows from the fact that $i\mathcal{O}$ allows witness encryption which allows NP secret sharing.

Note that existence of general $i\mathcal{O}$ implies existence of witness encryption [GGH$^+$13]. In witness encryption, the encryption has the message and an instance of a NP problem as inputs. The message is decrypted only if the decryption algorithm gets as in input a witness for that problem. With $i\mathcal{O}$ we can just obfuscate the circuit that outputs the message if it obtains a witness.

3.6.4 Outsourcing RAM

The setup for outsourcing is that we have a weak client that needs some computation results that it is unable to perform on its own. To still obtain those it outsources the computations to a possibly untrusted server. Hence, the client should do less work than is required for the computation and the private computation should be feasible on the server. A version of outsourcing private random-access machines (RAM) using $i\mathcal{O}$ was proposed in [GHRW14]. They consider reusable garbled RAM where the servers work depends on the RAM complexity of the computations and clients work is independent of the computation complexity.

Reusable garbled circuits, also extended to reusable garbled Turing Machines, enable to evaluate the garbled circuit many times with different inputs. The work [GHRW14] extends those ideas also to reusable garbled RAM by combining non-reusable garbled RAM and reusable garbled circuits. The core idea is to create a circuit that takes inputs as a randomness r and input x and, for a fixed RAM program, outputs a garbling of the RAM program using randomness r and the garbled inputs corresponding to x. The indistinguishability obfuscation is required to define a new version of reusable garbled circuits that satisfy security properties required for the garbled RAM. This is required as the previous constructions for reusable GC required the input size to depend on the

output size, but the circuits in this constructions have very large outputs. Essentially, the circuit is augmented to also validate the inputs and then obfuscated.

This solution can be applied also in multi-party settings where inputs originate from different parties. As practical future works it would be possible to either try to consider schemes that do not require $i\mathcal{O}$ or require weaker definitions of obfuscation than indistinguishability obfuscation.

3.7 Deniable Encryption

The idea of deniable public key encryption is that for an encryption for a message m and randomness r the source of this encryption can also provide false message and randomness pairs m^* and r^* that would give the same ciphertext. The real and fake messages should be indistinguishable for any parties not holding the private key.

Based on punctured programming approach, [SW14] proposes a deniable encryption scheme that has two obfuscated programs as public keys. The first is an obfuscation of the encryption function and the second is an obfuscation of a faking algorithm. Based on a ciphertext c and a message m^* the faking algorithm outputs a suitable randomness r^*. For that to work the encryption functionality is built so that it searches for hidden messages in the randomness and if found, they directly specify the output. For example, the randomness r^* would encode the value c as a encryption output. Again, this construction would be straight forward using black-box obfuscation, but requires the puncturing approach applied to PRF for $i\mathcal{O}$. Differently from previous attempts at deniable encryption, it is possible to compute the fake pair without initial message and randomness. They call this *publicly deniable encryption*.

Deniable encryption following the same puncturing idea is also described in [GP14].

3.8 Digital Signatures

As yet another application, $i\mathcal{O}$ has been used to build fast and short digital signatures [RW14]. They apply the punctured PRF technique, that was already applied for obtainign digital signatures in [SW14]. Especially, their goal is to obtain faster signing than existing algorithms at the expense of potentially longer verification. In [SW14] the signing algorithm picks a random key for a PRF and signature is computed by evaluating the punctured PRF value on it. The verification key is an obfuscation of the program that recomputes the PRF function and verifies that the two outputs match. This scheme is secure against an attacker who has to choose the message that it tries to forge before learning the public parameters of the scheme. The goal of [RW14] is to obtain an efficient adaptively secure signature scheme. For that they need to build a two-part signature based on the prefix guessing technique. However, the general idea is the same as both of these signature parts use puncturable PRFs and the verification key is an obfuscation of the verification program that contains the secret key.

3.9 Random Oracle Model

The Random Oracle model (RO) is a common abstraction used in cryptographic proofs to consider hash functions as truly random functions. They are used quite a lot, despite the fact that there exists constructions secure in the RO model, but not with any hash function. The existence of $i\mathcal{O}$ allows both to construct specific hash functions for some schemes and prove further that for some cases such hash functions do not exist.

3.9.1 Universal Computational Extractors

Universal Computational Extractors (UCE) can be based on $i\mathcal{O}$. They are proposed as a replacement for random oracles in many applications [BHK13b]. However, the first attempt was flawed as the existence of $i\mathcal{O}$ implies that no concrete hash function could also satisfy the definitions of UCE [BFM14a]. The definition was revised in September 2014 version of [BHK13a] as well as possible restrictions were proposed in [BFM14a]. The main difference from the random oracle model is that distinguishing between the random oracle and the hash function should be complicated for any party who only has some limited information about the possible queries to the challenge. Especially, a party called source can interact with the challenge, but can only leak some limited information to the distinguisher, whereas the source is restricted in its behaviour.

In turn, instantiations of UCE can result in, for example, correlation-secure hash functions (also correlated-input hash functions or CIH) and universal one-way functions as shown in [BM14]. Especially, [BM14] obtains the first CIH construction in the standard model. In addition, it validates that the updated definitions of UCE can be achieved by some hash functions. They follow the punctured programs idea of [SW14] and construct a hash function from punctured pseudo-random function that is obfuscated. They combine $i\mathcal{O}$ with auxiliary input point function obfuscation.

3.9.2 Full Domain Hash Signatures

Full domain hash signatures are a method to build a signature scheme from a trapdoor permutation and random oracle heuristic. FDHS are usually proven to be secure in random oracle model, however, using $i\mathcal{O}$ it is possible to build a suitable hash function [HSW14]. Especially, they develop a method to replace the specific way that the random oracle is programmed in these constructions with a family of full domain hash functions based on $i\mathcal{O}$ and punctured PRF like in [SW14]. Hence, they obtain the first known construction where the scheme in the random oracle model can be instantiated with a real family of hash function in the standard model without any modifications in the original scheme. Their hash functions are obfuscations of the programs that compute permutations of the output of a pseudo-random function in the input text.

3.9.3 Random Oracle Uninstantiability

As a contrary result from those of Sec 3.9.2 it is also shown that the existence of $i\mathcal{O}$ implies that some constructions in the random oracle model can not be instantiated

in the standard model [BFM14b]. Their work continues that of the UCEs [BFM14a]. They show that it is not possible to instantiate the Encrypt-with-Hash scheme with a real hash function. Such scheme would convert a randomised public-key encryption into a deterministic encryption by obtaining the randomness by hashing the message. They also generalise this to other transformations and schemes, for example turning CPA security to CCA security and turning authenticated encryption into key-dependant message resistant encryption.

As a similar result [BCPR14] shows that the existence of $i\mathcal{O}$ means that certain extractable one-way functions can not exist. This also invalidates the instantiability of the RO model as these functions exist if we assume the random oracle.

More general results are obtained by [GKMZ14]. As a concrete example they focus on bit-encryption and show a scheme secure in the random oracle model, but not with any concrete hash function. This public key bit-encryption is built from an existing IND-CPA secure bit-encryption and obfuscated circuit that encodes $y_i = h(x_i)$ and the secret key of the encryption scheme. The idea is that on an input of the correct hash function h, the circuit outputs the secret key. In the random oracle model, it is unlikely that such function exists, on the other hand, this function h is publicly known in the standard model. Therefore in the standard model such scheme can not be secure with any real hash function, as the secret key is always leaked. On the more general side, they show how to extend this approach to other schemes in the random oracle and also to provide separations for other idealised models.

3.10 Inconsistencies with Assumptions

In Sec 3.9 we saw that $i\mathcal{O}$ implies that some constructions in the Random Oracle model can not be instantiated with any constructions in the standard model. In this section we briefly consider other results that show how the existence of $i\mathcal{O}$ affects other previously made assumptions.

3.10.1 Black-box obfuscation

Interestingly, the existence of $i\mathcal{O}$ for general circuits implies several impossibilities for virtual black-box obfuscation [BCC$^+$14]. Although impossible in general, there have been some specific functionalities for which the virtual black-box obfuscation also exists. However, the existence of $i\mathcal{O}$ reduces the function classes for which we can hope for virtual black-box obfuscation. Especially, $i\mathcal{O}$ existence means that *auxiliary input* VBB as well as universal simulator VBB are both impossible for function family with super-polynomial pseudo-entropy. In the simulator case the adversary is unable to differentiate between interacting with a obfuscated program versus interacting with a simulator that has a black-box access to the obfuscated program. For the auxiliary input VBB, the adversary has some prior knowledge regarding the circuit or obfuscation. In fact, these notions are equivalent and required when using the obfuscation in composition with other protocols. Roughly, this is the class of functions where it is hard to distinguish

whether a function is in the family or the function has been slightly modified and outside the family. Their results also use the puncturable PRF concept from [SW14].

3.10.2 Circular security

Circular security is a security notion connected to public-key cryptography. The adversary there is given a cycle of ciphertexts $\mathsf{Enc}_{pk_1}(sk_2) \ldots \mathsf{Enc}_{pk_n}(sk_1)$ for different keys, each where each public key is used to encrypt the secret key of the next keypair in the cycle. Using $i\mathcal{O}$, it was possible to establish that for any cycle length n there exist IND-CPA secure cryptosystems that are not secure with respect to circular security [KRW13, MO14]. The notion of circular security is important because secret keys are very special plaintexts.

Note that in fact, the results of [KRW13] also apply for bit-encryption where it is not clear how to encrypt the secret key at all. In addition, [KRW13] also shows that if there exists a cryptosystem such that the adversary can distinguish the key cycle from a cycle of encryptions of zeros, then for a transformed IND-CPA secure cryptosystem an attacker can learn the secret keys from the cycle.

Note that [MO14] at first show that if VBB would be possible then IND-CPA security would not imply circular security. However, they also extend this to the $i\mathcal{O}$ case using the puncture programming approach [SW14].

4 Conclusion

The definition of indistinguishability obfuscation is currently the best formalisation that we have for the problem of obfuscation. Further more, we know that each efficient obfuscator for this definition is also the best possible obfuscator that we might have. Therefore, in the future it may be that we come up with more easily usable definitions of obfuscation, but they will either be weaker than indistinguishability obfuscation or also concide with the best-possible obfuscation. Especially, in the latter case, we know that the obfuscators defined for the indistinguishability obfuscation definitions would also satisfy the new definition.

Despite the fact that indistinguishability obfuscation is in general considered tricky to be used in cryptographic schemes this paper listed numerous results on various topics of cryptography, where current state of the art has been improved thanks to the existence of indistinguishability obfuscators.

References

[ABF+13] Jol Alwen, Manuel Barbosa, Pooya Farshim, Rosario Gennaro, S.Dov Gordon, Stefano Tessaro, and DavidA. Wilson. On the relationship between functional encryption, obfuscation, and fully homomorphic encryption. In

Martijn Stam, editor, *Cryptography and Coding*, volume 8308 of *Lecture Notes in Computer Science*, pages 65–84. Springer Berlin Heidelberg, 2013.

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. `http://eprint.iacr.org/`.

[BCC+14] Nir Bitansky, Ran Canetti, Henry Cohn, Shafi Goldwasser, YaelTauman Kalai, Omer Paneth, and Alon Rosen. The impossibility of obfuscation with auxiliary input or a universal simulator. In JuanA. Garay and Rosario Gennaro, editors, *Advances in Cryptology CRYPTO 2014*, volume 8617 of *Lecture Notes in Computer Science*, pages 71–89. Springer Berlin Heidelberg, 2014.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *Theory of Cryptography*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73. Springer Berlin Heidelberg, 2014.

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 505–514, New York, NY, USA, 2014. ACM.

[BFM14a] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and uces: The case of computationally unpredictable sources. In JuanA. Garay and Rosario Gennaro, editors, *Advances in Cryptology CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 188–205. Springer Berlin Heidelberg, 2014.

[BFM14b] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Random oracle uninstantiability from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2014/867, 2014. `http://eprint.iacr.org/`.

[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 *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, pages 1–18, London, UK, UK, 2001. Springer-Verlag.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, May 2012.

[BHK13a] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via uces. Cryptology ePrint Archive, Report 2013/424, 2013. `http://eprint.iacr.org/`.

[BHK13b] Mihir Bellare, VietTung Hoang, and Sriram Keelveedhi. Instantiating random oracles via uces. In Ran Canetti and JuanA. Garay, editors, *Advances in Cryptology CRYPTO 2013*, volume 8043 of *Lecture Notes in Computer Science*, pages 398–415. Springer Berlin Heidelberg, 2013.

[BM14] Christina Brzuska and Arno Mittelbach. Using indistinguishability obfuscation via uces. Cryptology ePrint Archive, Report 2014/381, 2014. `http://eprint.iacr.org/`, to appear in ASIACRYPT 2014.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *Theory of Cryptography*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer Berlin Heidelberg, 2011.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In JuanA. Garay and Rosario Gennaro, editors, *Advances in Cryptology CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 480–499. Springer Berlin Heidelberg, 2014.

[CGP14] Ran Canetti, Shafi Goldwasser, and Oxana Poburinnaya. Adaptively secure two-party computation from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2014/845, 2014. `http://eprint.iacr.org/`.

[CLTV14] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. Cryptology ePrint Archive, Report 2014/882, 2014. `http://eprint.iacr.org/`.

[DSKR14] Dana Dachman-Soled, Jonathan Katz, and Vanishree Rao. Adaptively secure, universally composable, multi-party computation in constant rounds. Cryptology ePrint Archive, Report 2014/858, 2014. `http://eprint.iacr.org/`.

[GGG+14] Shafi Goldwasser, S.Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In PhongQ. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602. Springer Berlin Heidelberg, 2014.

[GGH+13] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 40–49, Oct 2013.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure mpc from indistinguishability obfuscation. In Yehuda Lindell, editor, *Theory of Cryptography*, volume 8349 of *Lecture Notes in Computer Science*, pages 74–94. Springer Berlin Heidelberg, 2014.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In JuanA. Garay and Rosario Gennaro, editors, *Advances in Cryptology CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 518–535. Springer Berlin Heidelberg, 2014.

[GGJS13] Shafi Goldwasser, Vipul Goyal, Abhishek Jain, and Amit Sahai. Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/727, 2013. `http://eprint.iacr.org/`.

[GHRW14] Craig Gentry, Shai Halevi, Mariana Raykova, and Daniel Wichs. Outsourcing private ram computation. Cryptology ePrint Archive, Report 2014/148, 2014. `http://eprint.iacr.org/`.

[GJKS13] Vipul Goyal, Abhishek Jain, Venkata Koppula, and Amit Sahai. Functional encryption for randomized functionalities. Cryptology ePrint Archive, Report 2013/729, 2013. `http://eprint.iacr.org/`.

[GKL+13] S. Dov Gordon, Jonathan Katz, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/774, 2013. `http://eprint.iacr.org/`.

[GKMZ14] Matthew D. Green, Jonathan Katz, Alex J. Malozemoff, and Hong-Sheng Zhou. A unified approach to idealized model separations via indistinguishability obfuscation. Cryptology ePrint Archive, Report 2014/863, 2014. `http://eprint.iacr.org/`.

[GP14] Sanjam Garg and Antigoni Polychroniadou. Two-round adaptively secure mpc from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2014/844, 2014. `http://eprint.iacr.org/`.

[GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In *Proceedings of the 4th Conference on Theory of Cryptography*, TCC'07, pages 194–213, Berlin, Heidelberg, 2007. Springer-Verlag.

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In PhongQ. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 201–220. Springer Berlin Heidelberg, 2014.

[HW14] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. Cryptology ePrint Archive, Report 2014/669, 2014. http://eprint.iacr.org/.

[KNY14] Ilan Komargodski, Moni Naor, and Eylon Yogev. Secret-sharing for np. Cryptology ePrint Archive, Report 2014/213, 2014. http://eprint.iacr.org/.

[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/.

[LP14] Huijia Lin and Rafael Pass. Succinct garbling schemes and applications. Cryptology ePrint Archive, Report 2014/766, 2014. http://eprint.iacr.org/.

[MO14] Antonio Marcedone and Claudio Orlandi. Obfuscation $\rightarrow$ (ind-cpa security $\nrightarrow$ circular security). In *SCN*, pages 77–90, 2014.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/.

[PPS13] Omkant Pandey, Manoj Prabhakaran, and Amit Sahai. Obfuscation-based non-black-box simulation and four message concurrent zero knowledge for np. Cryptology ePrint Archive, Report 2013/754, 2013. http://eprint.iacr.org/.

[RW14] Kim Ramchen and Brent Waters. Fully secure and fast signing from obfuscation. Cryptology ePrint Archive, Report 2014/523, 2014. http://eprint.iacr.org/.

[SW08] Amit Sahai and Brent Waters. Slides on functional encryption. http://www.cs.utexas.edu/~bwaters/presentations/files/functional.ppt, 2008.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 475–484, New York, NY, USA, 2014. ACM.

[Wat14] Brent Waters. A punctured programming approach to adaptively secure functional encryption. Cryptology ePrint Archive, Report 2014/588, 2014. http://eprint.iacr.org/.

[YYHK14] Takashi Yamakawa, Shota Yamada, Goichiro Hanaoka, and Noboru Kunihiro. Self-bilinear map on unknown order groups from indistinguishability obfuscation and its applications. In JuanA. Garay and Rosario Gennaro, editors, *Advances in Cryptology CRYPTO 2014*, volume 8617 of *Lecture Notes in Computer Science*, pages 90–107. Springer Berlin Heidelberg, 2014.