Proceedings of the 2014 ACM Symposium on Theory of Computing

How to Use Indistinguishability Obfuscation: Deniable Encryption, and More

Amit Sahai[*]
UCLA
sahai@cs.ucla.edu

Brent Waters[†]
University of Texas at Austin
bwaters@cs.utexas.edu

We introduce a new technique, that we call *punctured programs*, to apply indistinguishability obfuscation towards cryptographic problems. We use this technique to carry out a systematic study of the applicability of indistinguishability obfuscation to a variety of cryptographic goals. Along the way, we resolve the 16-year-old open question of *Deniable Encryption*, posed by Canetti, Dwork, Naor, and Ostrovsky in 1997: In deniable encryption, a sender who is forced to reveal to an adversary both her message and the randomness she used for encrypting it should be able to convincingly provide "fake" randomness that can explain any alternative message that she would like to pretend that she sent. We resolve this question by giving the first construction of deniable encryption that *does not require any pre-planning* by the party that must later issue a denial.

In addition, we show the generality of our punctured programs technique by also constructing a variety of core cryptographic objects from indistinguishability obfuscation and one-way functions (or close variants). In particular we obtain: public key encryption, short "hash-and-sign" selectively secure signatures, chosen-ciphertext secure public key encryption, non-interactive zero knowledge proofs (NIZKs), injective trapdoor functions, and oblivious transfer. These results suggest the possibility of indistinguishability obfuscation becoming a "central hub" for cryptography.

[*]Research supported in part from a DARPA/ONR PROCEED award, NSF grants 1228984, 1136174, 1118096, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11- 1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

[†]Research supported in part from NSF CNS-0915361 and CNS-0952692, CNS-1228599 DARPA through the U.S. Office of Naval Research under Contract N00014-11-1-0382, DARPA N11AP20006, Google Faculty Research award, the Alfred P. Sloan Fellowship, Microsoft Faculty Fellowship, and Packard Foundation Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense or the U.S. Government.

Categories and Subject Descriptors

F.1.3 [**COMPUTATION BY ABSTRACT DEVICES**]: Complexity Measures and Classes—*Reducibility and completeness*

General Terms

Security; Theory

Keywords

Obfuscation, Encryption

1. INTRODUCTION

In 2001, Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang [1, 2] initiated the systematic study of *program obfuscation*, which aims to make computer programs "unintelligible" while preserving their functionality. They offered two notions of general obfuscation: First, they suggested a quite intuitive notion called *virtual black-box* obfuscation – which asks that an obfuscated program be no more useful than a black box implementing the program. Perhaps unsurprisingly, they showed that this notion of virtual black-box obfuscation would have many immediate and intuitive applications throughout cryptography, including for example, giving a transformation converting a natural private-key encryption scheme into a public-key encryption scheme. Unfortunately, however, they also showed that this notion of general-purpose virtual black-box obfuscation is impossible to achieve. Second, motivated by this impossibility, they also proposed the less intuitive, but potentially realizable, notion of *indistinguishability* obfuscation – which asks only that obfuscations of any two distinct (equal-size) programs that implement *identical* functionalities be computationally indistinguishable from each other. Unfortunately, it has been less clear how useful indistinguishability obfuscation would be. Recently, Garg, Gentry, Halevi, Raykova, Sahai, and Waters [12] proposed the first candidate construction of an efficient indistinguishability obfuscator for general programs (written as boolean circuits), and also showed how to apply indistinguishability obfuscation to achieve powerful new functional encryption schemes for general circuits.

We believe the work of [12] is likely to lead to significant follow-up research proposing alternative constructions of indistinguishability obfuscators under various other (possibly standard) assumptions, perhaps in a manner similar to research on Fully Homomorphic Encryption [14, 7, 6, 5, 15].

Despite this significant advance, some of the most basic questions about how to use indistinguishability obfuscation remain open. For example, can we use indistinguishability obfuscation to transform a natural private-key encryption scheme to a public-key encryption scheme, as can be done using virtual black-box obfuscation? In this work, we take a fresh look at the question of how to use indistinguishability obfuscation in general contexts. To this end, we introduce a new technique for applying indistinguishability obfuscation, that we call *punctured programs*. We use this technique to carry out a systematic study of the applicability of indistinguishability obfuscation to a variety of cryptographic goals. Along the way, we resolve the 16-year-old open question of *Deniable Encryption*, posed by Canetti, Dwork, Naor, and Ostrovsky in 1997 [8]: In deniable encryption, a sender who is forced to reveal to an adversary both her message and the randomness she used for encrypting it should be able to convincingly provide "fake" randomness that can explain any alternative message that she would like to pretend that she sent. We resolve this question by giving the first construction of deniable encryption that *does not require any pre-planning* by the party that must later issue a denial.

Punctured Programs.

At the heart of our results is a new technique for applying indistinguishability obfuscation that we call *punctured programs*: At a very high-level, the idea of the technique is to alter a program (which is to be obfuscated) by surgically removing a key element of the program, without which the adversary cannot win the security game it must play, but in a way that does not alter the functionality of the program. This is best illustrated by means of an example:

Consider the problem of transforming a natural private-key encryption scheme into a public-key encryption scheme, by means of obfuscation. This idea was first proposed by Diffie and Hellman in their original paper on public-key cryptography [9]. For example, consider the following simple and natural private-key encryption scheme: the secret key is the key K to a pseudo-random function (PRF). To encrypt a message m, the sender picks a random string r, and outputs $(r, \mathrm{PRF}(K, r) \oplus m)$.

How can we use indistinguishability obfuscation to convert this encryption scheme into a public-key scheme? Intuitively speaking, security requires that an adversary who is given a challenge ciphertext $c^* = (r^*, e^*)$ encrypting some message m^*, should not be able to recover m^*. As a warmup, consider the following (flawed) solution idea: Let a public key simply be an indistinguishability obfuscation of the encryption function itself, namely the function:

$$f_K(r, m) = \Big(r, \mathrm{PRF}(K, r) \oplus m\Big)$$

The central idea behind our approach is to observe that if it were possible to build the function f_K using a "punctured" PRF key which gave no information about $\mathrm{PRF}(K, r^*)$, but correctly defined the PRF at all other strings $r \neq r^*$, then even if the adversary *knew* this punctured PRF secret key, it would not be able to break the security of the challenge ciphertext. In fact, it is possible to construct such punctured PRF secret keys, for the original GGM construction of PRFs [17]. This was recently observed independently by [3, 4, 20].

All that remains is to argue that we can replace the obfuscation of the original function $f_K(r, m) = (r, \mathrm{PRF}(K, r) \oplus m)$ with the obfuscation of the punctured function, in a manner that is indistinguishable to the adversary. However, this is in fact false: Indeed, there is a trivial attack on this suggestion because given the challenge ciphertext (r^*, e^*), the adversary can simply query the obfuscated program on the input $(r^*, 0)$ and thereby learn $\mathrm{PRF}(K, r^*)$ and recover the message m^* from the challenge ciphertext. Thus, we can never hope to swap the obfuscation of the original function with the punctured function, because the adversary can *always* tell the difference. Indeed, an indistinguishability obfuscator guarantees indistinguishable obfuscated programs only when given two *functionally equivalent* input programs. However, as we just saw, the punctured program is clearly not equivalent to the original program, precisely because of the puncturing.

This brings us to the second main idea behind the punctured programs technique, which is crucial to hide where the puncturing takes place and allow the use of indistinguishability obfuscation: The idea is to find a way to "move" the puncturing to a location that is *never* functionally accessed by the program. Let us illustrate by returning to the example of building a public-key encryption scheme from a natural private-key encryption scheme. Consider a pseudorandom generator PRG that maps λ bits to 2λ bits, where λ is a security parameter. Now consider the following modified private-key encryption function:

$$f_K(r, m) = \Big(\mathrm{PRG}(r), \mathrm{PRF}\Big(K, \mathrm{PRG}(r)\Big) \oplus m\Big)$$

As before, let the public key be an indistinguishability obfuscation of this function. Now, however, a challenge ciphertext will look like $c^* = (\mathrm{PRG}(r^*), \mathrm{PRF}(K, \mathrm{PRG}(r^*)) \oplus m^*)$. Since a PRG is keyless, however, the adversary cannot distinguish the original security game in which the challenge ciphertext was created as above, and a hybrid experiment where the challenge ciphertext is created with a freshly chosen random $\tilde{r} \in \{0,1\}^{2\lambda}$, as follows: $c^* = (\tilde{r}, \mathrm{PRF}(K, \tilde{r}) \oplus m^*)$. However, now we observe that with overwhelming probability, $\tilde{r}$ is outside the image of the PRG. Thus, crucially, the encryption function f_K never makes use of the PRF evaluated at $\tilde{r}$.

Thus, indistinguishability obfuscation guarantees that an obfuscation of an alternative program that uses a punctured PRF key that carves out $\tilde{r}$ is indistinguishable from the obfuscation of the original program, because these two programs are functionally equivalent. Now, due to the puncturing, the adversary simply does not have enough information to distinguish $\mathrm{PRF}(K, \tilde{r})$ from random, and thus security follows.

This example illustrates the core ideas behind the punctured programs technique. While this example is quite simple, it already has surprising implications: For example, the scheme above would have slow encryption, but the speed of decryption would be as fast as evaluating a single call to the GGM PRF, where the PRG underlying the GGM construction could be instantiated using a standard symmetric-key primitive such as AES. Furthermore, the size of ciphetexts in the above scheme, if instantiated using the GGM construction applied to 128-bit AES, could be as short as 300 bits. No existing construction of public-key encryption that we are aware of enjoys such features.

We use this technique to use indistinguishability obfuscation to build several cryptographic primitives, including short signatures, CCA-secure encryption, and perfect non-interactive zero-knowledge arguments for NP. Many of these applications enjoy similar surprising features to the ones stated above for public-key encryption. Most notably, we use this technique to solve the open question of deniable encryption.

On the need for additional assumptions.

Before moving on to deniable encryption, we note that our construction of public-key encryption above required not only indistinguishability obfuscation, but also the assumption that one-way functions exist. It is natural to wonder if this extra assumption of one-way functions is needed. We observe that this indeed seems necessary: If P=NP, then certainly one-way functions would not exist and public-key encryption would be impossible; however, if P=NP then indistinguishability obfuscators *would* exist for all circuits. This follows immediately from the fact that computing the lexicographically first equivalent circuit to an input circuit is in the polynomial hierarchy (specifically in $P^{\Sigma_3^P}$). Thus, it should not be possible to construct interesting cryptographic functionalities from indistinguishability obfuscators alone without proving P≠NP along the way.

Subsequent to this work, Moran and Rosen [21] gave an elegant and simple argument that if NP $\neq$ co-RP, then $i\mathcal{O}$ actually implies the existence of one way functions.

Deniable Encryption.

In 1997, Canetti, Dwork, Naor, and Ostrovsky [8] proposed the intriguing problem of *deniable encryption*: In deniable (public-key) encryption, there is a public key used for encryption by an honest sender, to produce a ciphertext that is made public. However, at a later time, an adversary may approach the sender and force her to reveal the message that she encrypted together with the randomness that she used to encrypt the message. In deniable encryption, we would like to provide the sender with a "faking" algorithm that allows her to produce fake randomness for any message of her choice, that appears indistinguishable from the true randomness to any adversary (that does not possess the secret decryption key). Note that any deniable encryption scheme must incur at least some negligible decryption error. Unfortunately, all known proposed algorithms for deniable encryption allow an attacker to gain a noticeable $1/n$ advantage in distinguishing real randomness from fake randomness, where roughly speaking n is the length of the ciphertext [8]. In fact, over the last 16 years, high-profile papers have claimed solutions to this problem, most notably the recent work of [10], but these have all been shown to be incorrect [11].

We propose a completely new approach to deniable encryption based on obfuscation. The essential idea behind our approach is the following: The public key will consist of two obfuscated programs.

- The first obfuscated program Encrypt is used for encryption of ciphertexts. This program takes as input a message m and a long random string u. Before encrypting, it first checks for a special "hidden sparse trigger". It checks to see if the random coins are a special encoding from a sparse space. To do this it checks if the claimed randomness u is *itself* an encoding of some valid ciphertext c and the message m, under a (different internal) encoding scheme described below. If so, it outputs this ciphertext c. Otherwise, it uses u to encrypt m in an (essentially) normal way to produce a ciphertext c under a standard encryption E_{PKE}.
- The second program is an obfuscation of an Explain algorithm. The explanation algorithm takes as input a ciphertext c, a message m, and some randomness. This explanation algorithm doesn't care what c is "really" encrypting. It simply outputs an encoding of (c, m). Thus, the explanation mode can be used (by anyone) to create fake randomness that can explain any ciphertext as any desired message.

Thus, our approach uses two algorithms in tandem. The Encrypt algorithm will produce encryptions of some message m. When run with coins u chosen truly at random, it will almost always correctly encrypt m, since the "hidden sparse triggers" form a negligible fraction of possible u values.

The Explain algorithm takes in a ciphertext-message pair (c, m) and will produce an encoding u' that serves as a hidden trigger for the Encrypt program. We use the terminology "ciphertext" to refer to the ciphertext, produced by Encrypt, that will be sent to the receiver. We use "encoding" or "hidden trigger encoding" to refer to a value output by the Explain program. The intended semantics are that if Encrypt program discovers this hidden trigger, it is to immediately output the ciphertext c that is hidden in the encoding.

For this approach to work, clearly Explain will need to produce pseudorandom encodings, so that the adversary cannot immediately distinguish fake randomness from true randomness. Upon closer inspection, we see that the attacker can also take some randomness u^* given by the honest sender, modify it, and provide it as input to the Encrypt program, and see what happens. Indeed, we see that the encodings produced by Explain will in fact need to be CCA-secure, since the Encrypt program itself implements a limited kind of CCA attack on Explain encodings. For example, if the adversary can take a fake randomness u^*, and modify it slightly so that it still encodes the same ciphertext-message pair (c^*, m^*), then the adversary would most likely learn that the sender is faking. Note that we must show that encodings produced by Explain are pseudorandom, even when subject to this kind of CCA attack.

Finally, we must make our construction work given only an indistinguishability obfuscator, not a virtual black-box obfuscator. To do this, we again make use of the punctured programs technique with multiple PRFs that have been enhanced to have additional properties: we consider *injective* PRFs, where the function $\text{PRF}(K, \cdot)$ is injective with high probability over the choice of key K; and also what we call *extracting* PRFs, where the distribution $(K, \text{PRF}(K, X))$ is very close to a uniform distribution as long as the distribution X has sufficient min-entropy. We show how to build injective and extracting puncturable PRFs assuming only one-way functions. Thus, we obtain deniable encryption from indistinguishability obfuscation and one-way functions.

Publicly Deniable Encryption.

Our obfuscation-based construction of deniable encryption actually achieves a new and stronger notion that we call *publicly deniable encryption.* Recall that in a traditional deniable encryption scheme, the sender must remember his actual randomness and use it to construct fake randomness for other messages. In a publicly deniable encryption scheme, we do not require a sender to know the randomness that was used to generate any particular ciphertext in order to be able to generate fake randomness for it. Thus, even a third party that had nothing to do with the actual honest generation of a ciphertext can nonetheless produce fake randomness for that ciphertext corresponding to any message of his choice. Note that our construction idea above achieves this, since the explanation mode does not need to see what randomness was originally used to create a particular ciphertext.

The public deniability aspect of our security notion allows us to think of producing fake randomness as a kind of "simulation." With this perspective, we observe that we can separate out the security of publicly deniable encryption into two parts: First, we must prove ordinary indistinguishability (IND-CPA) security of the encryption scheme. In other words, an encryption of m_0 must be indistinguishable from an encryption of m_1 for any two messages m_0, m_1. Second, we have a separate "explanation" security requirement, that says that for any fixed message m, an encryption of m together with m and the true randomness used to create that ciphertext should be indistinguishable from an encryption of m together with the *same* message m but with fake randomness explaining the ciphertext (also as an encryption of m).

Note, however, that these two security properties together imply ordinary deniability, since one can first invoke explanation security to create a hybrid where a fake explanation is offered with respect to the actual message m_b that was encrypted, and then invoke IND-CPA to argue that this must be indistinguishable from a fake explanation offered about a ciphertext that was actually created using m_{1-b}.

Universal Deniable Encryption.

We also consider another new notion that we call *universal deniable encryption.* In universal deniable encryption, we imagine that there are several different public key infrastructures (PKIs) already set up for a variety of legacy public-key encryption schemes (perhaps such as IND-CPA-secure systems based on RSA or El-Gamal). However, we would like to enable users to deniably encrypt messages, without requiring all parties to rekey their public keys. We can do so as long as there is a trusted party (that cannot be coerced into revealing its randomness by the adversary) that is able to publish a single additional public parameter. (The trusted party plays no further role.) Users are now instructed to encrypt all messages using a combination of the public parameters and the recipient's legacy public key. A recipient will still be able to decrypt the message using its original secret key and decryption algorithm. If an adversary now approaches a user and forces the user to reveal what message and randomness it used (knowing that this user would have used the new standard encryption mechanism to encrypt), the user can provide convincing fake randomness to the adversary to explain any message of his choice.

Technical overview of Deniable Encryption Scheme.

We now provide a brief overview of how our main publicly deniable encryption scheme works (this overview does not refer to universal deniable encryption). The technical heart of our construction is the $\mathsf{Explain}$ encoding scheme, and the argument that encodings produced by this encoding scheme on input (c^*, m^*) are indistinguishable from true randomness used to produce c^* when encrypting m^* using E_{PKE}, even when the adversary is given access to the obfuscated $\mathsf{Encrypt}$ and $\mathsf{Explain}$ programs. The construction and argument crucially relies on the punctured programs technique.

We think of the randomness u used by $\mathsf{Encrypt}$ as consisting of two strings $u[1]$ and $u[2]$. To create a fake explanation that a ciphertext c is an encryption of a bit m, $\mathsf{Explain}$ will use additional randomness r to produce an encoding as follows: Set $\alpha = \mathrm{PRF}_2(K_2, (m, c, \mathrm{PRG}(r)))$. Let $\beta = \mathrm{PRF}_3(K_3, \alpha) \oplus (m, c, \mathrm{PRG}(r))$. The encoding e will be $e = (\alpha, \beta)$. Here, it is important that PRF_2 is an injective puncturable PRF with a large co-domain, and that PRF_3 is a puncturable PRF.

Consider a ciphertext $c^* = E_{\mathrm{PKE}}(m^*, u^*)$, and a particular fake explanation encoding $e^* = (\alpha^*, \beta^*)$ for (c^*, m^*), that was generated using randomness r^*. We will show that e^* is "pseudorandom" by repeated application of the punctured programs technique, by which eventually we will replace e^* with a random string (such that it always yields a particular ciphertext c^* when applied on input m^* in $\mathsf{Encrypt}$). Recall that the key technical difficulty in applying the punctured programs technique is to ensure that when puncturing a program – cutting some potential computation out of the program – we must make sure that the *functionality* of the program is not affected, so that we may apply indistinguishability obfuscation. The essential way we do this with respect to the $\mathsf{Explain}$ procedure described above is:

1. First, we replace $\mathrm{PRG}(r^*)$ with a totally random value $\tilde{r}$, so that we can argue that $\mathrm{PRF}_2(K_2, \cdot)$ will never be invoked on $(m^*, c^*, \tilde{r})$ in the functionality of $\mathsf{Explain}$. This lets us puncture out $(m^*, c^*, \tilde{r})$ from the PRF key K_2.

2. Having punctured out $(m^*, c^*, \tilde{r})$, we can then replace α^* with a completely random string, because of the punctured PRF key. By the sparseness of the image of $\mathrm{PRF}_2(K_2, \cdot)$, because it is injective, this means that we can then safely puncture α^* from the PRF key K_3 without disturbing the functionality of $\mathsf{Explain}$.

3. Finally, then, we are able to replace β^* with a random value.

Above we sketched the changes to $\mathsf{Explain}$, but we note that additional (even greater) changes will need to be made to $\mathsf{Encrypt}$, because this procedure must maintain the invariant that $\mathsf{Encrypt}(m^*; u^*) = \mathsf{Encrypt}(m^*; e^*) = c^*$ throughout all our changes. This proceeds in a manner similar to what is described above, but is somewhat more technical, and we crucially use the injectivity of $\mathrm{PRF}(K_2, \cdot)$ beyond just the sparseness of its image.

Hidden Sparse Triggers.

We briefly recap the main properties of our hidden sparse trigger concept as we believe that it will have applications elsewhere in cryptography. The highlighted properties are:

- Sparseness. The trigger values must live in a sparse subset of some larger set. For our application, if the triggers were not sparse the deniable encryption system would not be correct.

- Oblivious Sampling. It must be possible to sample obliviously from the full set. In our construction, this is simply choosing a string uniformly at random.

- Indistinguishability under malleability attacks. It should be hard to distinguish a value chosen randomly from the larger set from the sparse hidden trigger set. This must be true even given a program that detects and utilizes the hidden trigger.

- Publicly Generated Triggers. It should be possible for anyone to generate hidden triggers.

Core primitives from indistinguishability obfuscation. We show the generality of our punctured programs technique by also constructing a variety of core cryptographic objects from indistinguishability obfuscation and one-way functions (or close variants). In particular, we obtain:

- Public-key encryption from indistinguishability obfuscation and one-way functions, as already sketched in the introduction above.

- Short "hash-and-sign" selectively secure signatures from indistinguishability obfuscation and one-way functions. These signatures are secure in a model where the adversary must declare the message m^* on which a forgery will take place at the start, but then can adaptively query any number of messages distinct from m^* on which it can receive signatures.

- An adaptively chosen-ciphertext secure (CCA2-secure) public-key encryption scheme from indistinguishability obfuscation and one-way functions.

- An adaptively chosen-ciphertext secure (CCA2-secure) key encapsulation mechanism (KEM) from indistinguishability obfuscation and one-way functions. This construction is remarkably simple.

- Perfect non-interactive zero knowledge (NIZK) arguments for all NP languages in the common reference string (CRS) model from indistinguishability obfuscation and one-way functions.

- Injective trapdoor functions from indistinguishability obfuscation and injective one-way functions.

- Oblivious Transfer protocols from indistinguishability obfuscation and one-way functions.

Taken altogether these results (along with the functional encryption result of GGHRSW [12]) cover a large swathe of cryptography. This feeds into our research vision of Indistinguishability Obfuscation becoming a central hub of cryptography. In an ideal future "most" cryptographic primitives will be shown to be realizable from $i\mathcal{O}$ and one way functions.

Organization. As this is only an extended abstract, we defer a formal treatment of most results and proofs to our full version [22].

2. PUBLICLY DENIABLE ENCRYPTION

We now give our definition of publicly deniable encryption. We show that our notion of publicly deniable encryption implies the notion of deniable encryption of [8] (without any "plan-ahead" requirement).

DEFINITION 1 (PUBLICLY DENIABLE ENCRYPTION). *A publicly deniable encryption scheme over a message space $\mathcal{M} = \mathcal{M}_\lambda$ consists of four algorithms* Setup $\mathsf{Encrypt}$, $\mathsf{Decrypt}$, $\mathsf{Explain}$.

- $\mathsf{Setup}(1^\lambda)$ *is a polynomial time algorithm that takes as input the unary representation of the security parameter λ and outputs a public key* PK *and a secret key* SK.

- $\mathsf{Encrypt}(\mathrm{PK}, m \in \mathcal{M}; u)$ *is a polynomial time algorithm that takes as input the public key* PK *a message m and uses random coins u. (We call out the random coins explicitly for this algorithm). It outputs a ciphertext c.*

- $\mathsf{Decrypt}(\mathrm{SK}, c)$ *is a polynomial time algorithm that takes as input a secret key* SK *and ciphertext c, and outputs a message m.*

- $\mathsf{Explain}(\mathrm{PK}, c, m; r)$ *is a polynomial time algorithm that takes as input a public key* PK, *a ciphertext c, and a message m, and outputs and a string e, that is the same size as the randomness u taken by* $\mathsf{Encrypt}$ *above.*

We say that a publicly deniable encryption scheme is correct if for all messages $m \in \mathcal{M}$:

$$\Pr\left[\begin{array}{c}(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Setup}(1^\lambda);\\ \mathsf{Decrypt}(\mathrm{SK}, \mathsf{Encrypt}(\mathrm{PK}, m; u)) \neq m\end{array}\right] = negl(\lambda)$$

where $negl(\cdot)$ is a negligible function.

REMARK 1. *The* $\mathsf{Explain}$ *algorithm does not directly factor into the correctness definition.*

2.1 Security

Any publicly deniable encryption system must meet two security requirements both given as game-based definitions. The first is Indistinguishability under Chosen Plaintext Attack (IND-CPA). This is exactly the same security game [18] as in standard public key encryption schemes. (We include a description here for self-containment.)

The second is a new security property called indistinguishability of *explanation*. Informally speaking, if one considers a message m, an encryption c of m and the coins, u used in encryption, then indistinguishability of explanation states that no attacker can distinguish between (m, c, u) and $(m, c, \mathsf{Explain}(c, m; r))$. We now provide both security definitions formally:

Indistinguishability under Chosen Plainext Attack.

We describe the IND-CPA game as a multi-phase game between an attacker $\mathcal{A}$ and a challenger.

Setup: The challenger runs $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Setup}(1^\lambda)$ and gives PK to $\mathcal{A}$.

Challenge: $\mathcal{A}$ submits two messages $m_0, m_1 \in \mathcal{M}$, and is given $c^* = \mathsf{Encrypt}(\mathrm{PK}, m_b; u)$ for random coins u.

Guess $\mathcal{A}$ then outputs a bit b' in $\{0, 1\}$.

We define the advantage of an algorithm $\mathcal{A}$ in this game to be

$$\mathsf{Adv}_{\mathcal{A}} = \Pr[b' = b] - \frac{1}{2}$$

Definition 2. *An publicly deniable scheme is IND-CPA secure if for all poly-time $\mathcal{A}$ the function $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ is negligible.*

Indistinguishability of Explanation.

We describe the Indistinguishability of Explanation security game as a multi-phased game between an $\mathcal{A}$ and a challenger.

Setup: The challenger runs $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Setup}(1^{\lambda})$ and gives PK to $\mathcal{A}$.

Challenge: $\mathcal{A}$ submits *a single* message $m \in \mathcal{M}$. The challenger creates $c^* = \mathsf{Encrypt}(\mathrm{PK}, m; u)$ for random u and it computes $e = \mathsf{Explain}(\mathrm{PK}, c^*; r)$ for random r. The challenger finally flips a coin $b \in \{0, 1\}$. If $b = 0$ it outputs (c^*, u). If $b = 1$ it outputs (c^*, e).

Guess $\mathcal{A}$ then outputs a bit b' in $\{0, 1\}$.

We define the advantage of an algorithm $\mathcal{A}$ in this game to be

$$\mathsf{Adv}_{\mathcal{A}} = \Pr[b' = b] - \frac{1}{2}$$

Definition 3. *An publicly deniable scheme has Indistinguishability of Explanation if for all poly-time $\mathcal{A}$ the function $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ is negligible.*

Remark 2. *We remark that a publicly deniable encryption scheme for one bit messages immediately implies a publicly deniable encryption scheme for multi-bit messages of any polynomial length n bits. One can simply create a ciphertext for n bit message as a sequence of n single bit encryptions (i.e. "bit-by-bit encryption"). The* Explain *algorithm for the n bit scheme simply calls the single bit explain algorithm for each of the n ciphertext components. Security for both IND-CPA and Indistinguishability of Explanation follows via the usual one bit to many bit hybrid argument.*

2.2 Implication for Deniable Encryption

We now show that a publicly deniable encryption is also a deniable encryption as defined by CDNO [8]. We begin by briefly recalling the CDNO notion of (sender) deniable encryption. We simplify the exposition to consider encryption as a standard one pass algorithm that takes in the public key and outputs a ciphertext.

A (standard) deniable encryption scheme has four algorithms: $\mathsf{Setup}_{\mathsf{DE}}, \mathsf{Encrypt}_{\mathsf{DE}}, \mathsf{Decrypt}_{\mathsf{DE}}, \mathsf{Fake}_{\mathsf{DE}}$.

- $\mathsf{Setup}_{\mathsf{DE}}(1^{\lambda})$ is a polynomial time algorithm that takes as input the unary representation of the security parameter λ and outputs a public key PK and a secret key SK.
- $\mathsf{Encrypt}_{\mathsf{DE}}(\mathrm{PK}, m \in \mathcal{M}; u)$ is a polynomial time algorithm that takes as input the public key PK a message m and uses random coins u. (We call out the random coins explicitly for this algorithm). It outputs a ciphertext c.
- $\mathsf{Decrypt}_{\mathsf{DE}}(\mathrm{SK}, c)$ is a polynomial time algorithm that takes as input a secret key SK and ciphertext c, and outputs a message m.
- $\mathsf{Fake}_{\mathsf{DE}}(\mathrm{PK}, m, u, c, m'; r)$ is a polynomial time algorithm that takes as input a public key PK, an "original" message m, bitstring u, ciphertext c, and a "faked" message m', and outputs and a string e, that is the same size as the randomness u taken by $\mathsf{Encrypt}$ above.

We can see that the algorithms essentially align with the publicly deniable algorithms. The main exception is that the $\mathsf{Fake}_{\mathsf{DE}}$ algorithm takes in the original randomness used for encryption as well as the original message. Whereas the $\mathsf{Explain}$ algorithm of publicly deniable encryption only requires the transmitted ciphertext and new message (and none of the other history.)

Security in standard deniable encryption consists of two notions. The first is IND-CPA as given above for publicly deniable encryption. The second is a deniability game given here.

Setup: The challenger runs $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Setup}_{\mathsf{DE}}(1^{\lambda})$ and gives PK to $\mathcal{A}$.

Challenge: $\mathcal{A}$ submits *two* message $m_0, m_1 \in \mathcal{M}$. The challenger flips a random coin b. If $b = 0$, it creates $c^* = \mathsf{Encrypt}_{\mathsf{DE}}(\mathrm{PK}, m_0; u)$ for random u. It outputs (c^*, u).

If $b = 1$ it creates $c^* = \mathsf{Encrypt}_{\mathsf{DE}}(\mathrm{PK}, m_0; u)$ and it and it computes $e = \mathsf{Explain}(\mathrm{PK}, m_1, u, c^*, m_0; r)$ for random r. It outputs (c^*, e).

Guess $\mathcal{A}$ then outputs a bit b' in $\{0, 1\}$.

We define the advantage of an algorithm $\mathcal{A}$ in this game to be

$$\mathsf{Adv}_{\mathcal{A}} = \Pr[b' = b] - \frac{1}{2}$$

Definition 4. *A deniable encryption scheme is Deniable if for all poly-time $\mathcal{A}$ the function $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ is negligible.*

The Implication.

Now we can show that publicly deniable encryption implies standard deniable encryption. (We use the subscript of PDE to denote the original deniable encryption system and DE to denote the constructed deniable encryption system.) The first three algorithms are set to be exactly the same. $\mathsf{Setup}_{\mathsf{DE}} = \mathsf{Setup}_{\mathsf{PDE}}, \mathsf{Encrypt}_{\mathsf{DE}} = \mathsf{Encrypt}_{\mathsf{PDE}}, \mathsf{Decrypt}_{\mathsf{DE}} = \mathsf{Decrypt}_{\mathsf{PDE}}$. The faking algorithm is $\mathsf{Fake}_{\mathsf{DE}}(\mathrm{PK}, m, u, c, m'; r)$ $= \mathsf{Explain}(\mathrm{PK}, c, m')$. In essence, the Explain algorithm is simply used to fake and just doesn't bother to use the original randomness or message.

We now sketch by a simple hybrid argument that a deniable encryption scheme built from publicly deniable one (in the manner outlined above) is secure in the deniability game.

Let Hybrid:1, be the experiment where ($c^* = \mathsf{Encrypt_{DE}}$ $(\mathrm{PK}, m_0; u), u$) is output as an encryption of m_0 with the real coins. Now let Hybrid:2 be the experiment where ($CT^* = \mathsf{Encrypt_{DE}}(\mathrm{PK}, m_0; u), e = \mathsf{Explain_{PDE}}(\mathrm{PK}, c^*, m_0; r)$). By definition of the Indistinguishability of Explanation, Hybrid:1 is indistinguishabile from Hybrid:2.

Now let Hybrid:3 be the experiment where the challenger outputs ($c^* = \mathsf{Encrypt_{DE}}(\mathrm{PK}, m_1; u)$, $e = \mathsf{Explain_{PDE}}(\mathrm{PK}, c^*, m_0; r)) = \mathsf{Fake_{DE}}(\mathrm{PK}, m_1, u, c, m_0; r)$. Any attacker that can distinguish between Hybrid:1, Hybrid:2 can break IND-CPA of the scheme. A reduction algorithm $\mathcal{B}$ will simply pass the IND-CPA challenge ciphertext c^* of the attacker along with computing itself $e = \mathsf{Explain_{PDE}}(\mathrm{PK}, c^*, m_0; r)$). It then simply relays the attacker's answer to the IND-CPA game. The simplicity of this argument derives from the fact that the $\mathsf{Explain_{PDE}}$ algorithm *does not use* any of the coins used to create the original ciphertext.

We conclude by observing that Hybrid:1 and Hybrid:3 correspond respectively to where $b = 0$ and $b = 1$ in the CDNO game. Thus, a publicly deniable encryption scheme gives one in the standard sense.

Advantages of Publicly Deniable Encryption.

The abstraction of Publicly Deniable Encryption has two advantages. First, public deniability can be a useful property in practice. For example, consider a party was expected to reveal coins for a ciphertext, but had lost them. Using the public deniability $\mathsf{Explain}$ algorithm, it could make them up just knowing the transmitted ciphertext.

Second, in proving our construction of Section 4 the we found that proving indistinguishability of explanation is significantly simpler than directly proving deniability. [1] The primary reason stems from the fact that the indistinguishability of explanation only requires a single message in its security game. Our abstraction separates the (in)ability to distinguish between encryptions of different messages from the (in)ability to distinguish between legitimate and explained randomness. [2]

3. PRELIMINARIES

In this section, we define indistinguishability obfuscation, and variants of pseudo-random functions (PRFs) that we will make use of. All the variants of PRFs that we consider will be constructed from one-way functions.

Indistinguishability Obfuscation.

The definition below is from [12]; there it is called a "family-indistinguishable obfuscator", however they show that this notion follows immediately from their standard definition of indistinguishability obfuscator using a non-uniform argument.

Definition 5. (Indistinguishability Obfuscator ($i\mathcal{O}$)). *A uniform PPT machine $i\mathcal{O}$ is called an* indistinguishability obfuscator *for a circuit class $\{\mathcal{C}_\lambda\}$ if the following conditions are satisfied:*

- *For all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, for all inputs x, we have that*
$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\lambda, C)] = 1$$
- *For any (not necessarily uniform) PPT adversaries Samp, D, there exists a negligible function α such that the following holds: if $\Pr[\forall x,\ C_0(x) = C_1(x) : (C_0, C_1, \sigma) \leftarrow Samp(1^\lambda)] > 1 - \alpha(\lambda)$, then we have:*
$$\left| \begin{array}{l} \Pr\left[D(\sigma, i\mathcal{O}(\lambda, C_0)) = 1 : (C_0, C_1, \sigma) \leftarrow Samp(1^\lambda)\right] \\ -\Pr\left[D(\sigma, i\mathcal{O}(\lambda, C_1)) = 1 : (C_0, C_1, \sigma) \leftarrow Samp(1^\lambda)\right] \end{array} \right| \\ \leq \alpha(\lambda)$$

In this paper, we will make use of such indistinguishability obfuscators for all polynomial-size circuits:

Definition 6. (Indistinguishability Obfuscator for *P/poly*). *A uniform PPT machine $i\mathcal{O}$ is called an* indistinguishability obfuscator *for P/poly if the following holds: Let $\mathcal{C}_\lambda$ be the class of circuits of size at most λ. Then $i\mathcal{O}$ is an indistinguishability obfuscator for the class $\{\mathcal{C}_\lambda\}$.*

Such indistinguishability obfuscators for all polynomial-size circuits were constructed under novel algebraic hardness assumptions in [12].

PRF variants.

We first consider some simple types of constrained PRFs [3, 4, 20], where a PRF is only defined on a subset of the usual input space. We focus on *puncturable* PRFs, which are PRFs that can be defined on all bit strings of a certain length, except for any polynomial-size set of inputs:

Definition 7. *A* puncturable *family of PRFs F mapping is given by a triple of Turing Machines* Key_F, $\mathrm{Puncture}_F$, *and* Eval_F, *and a pair of computable functions* $n(\cdot)$ *and* $m(\cdot)$, *satisfying the following conditions:*

- ***[Functionality preserved under puncturing]*** *For every PPT adversary A such that $A(1^\lambda)$ outputs a set $S \subseteq \{0,1\}^{n(\lambda)}$, then for all $x \in \{0,1\}^{n(\lambda)}$ where $x \notin S$, we have that:*
$$\Pr\begin{bmatrix} \mathsf{Eval}_F(K, x) = \mathsf{Eval}_F(K_S, x) : \\ K \leftarrow \mathrm{Key}_F(1^\lambda), \\ K_S = \mathrm{Puncture}_F(K, S) \end{bmatrix} = 1.$$
- ***[Pseudorandom at punctured points]*** *For every PPT adversary (A_1, A_2) such that $A_1(1^\lambda)$ outputs a set $S \subseteq \{0,1\}^{n(\lambda)}$ and state σ, consider an experiment where $K \leftarrow \mathrm{Key}_F(1^\lambda)$ and $K_S = \mathrm{Puncture}_F(K, S)$. Then we have*
$$\left| \begin{array}{l} \Pr\left[A_2(\sigma, K_S, S, \mathsf{Eval}_F(K, S)) = 1\right] - \\ \Pr\left[A_2(\sigma, K_S, S, U_{m(\lambda)\cdot|S|}) = 1\right] \end{array} \right| = negl(\lambda)$$
where $\mathsf{Eval}_F(K, S)$ denotes the concatenation of Eval_F $(K, x_1)), \ldots, \mathsf{Eval}_F(K, x_k))$ where $S = \{x_1, \ldots, x_k\}$ is the enumeration of the elements of S in lexicographic order, $negl(\cdot)$ is a negligible function, and U_ℓ denotes the uniform distribution over ℓ bits.

For ease of notation, we write $F(K, x)$ to represent Eval_F (K, x). *We also represent the punctured key* $\mathrm{Puncture}_F(K, S)$ *by $K(S)$.*

[1]In an earlier internal draft we proved deniability directly; however, the proof was more complex and required an additional property on the constrained PRFs that was not obtainable from one way functions.

[2] We shall see that even if an attacker is given the encryption systems secret key, it cannot win in the explanation game.

The GGM tree-based construction of PRFs [17] from one-way functions are easily seen to yield puncturable PRFs, as recently observed by [3, 4, 20]. Thus we have:

THEOREM 1. *[17, 3, 4, 20] If one-way functions exist, then for all efficiently computable functions $n(\lambda)$ and $m(\lambda)$, there exists a puncturable PRF family that maps $n(\lambda)$ bits to $m(\lambda)$ bits.*

Next we consider families of PRFs that are with high probability injective:

DEFINITION 8. *A* statistically injective *(puncturable) PRF family with failure probability $\epsilon(\cdot)$ is a family of (puncturable) PRFs F such that with probability $1 - \epsilon(\lambda)$ over the random choice of key $K \leftarrow \mathrm{Key}_F(1^\lambda)$, we have that $F(K, \cdot)$ is injective.*

If the failure probability function $\epsilon(\cdot)$ is not specified, then $\epsilon(\cdot)$ is a negligible function.

THEOREM 2. *If one-way functions exist, then for all efficiently computable functions $n(\lambda)$, $m(\lambda)$, and $e(\lambda)$ such that $m(\lambda) \geq 2n(\lambda) + e(\lambda)$, there exists a puncturable statistically injective PRF family with failure probability $2^{-e(\lambda)}$ that maps $n(\lambda)$ bits to $m(\lambda)$ bits.*

PROOF. For ease of notation, we suppress the dependence of n, m, e on λ. Let $\mathcal{H}$ be a family of 2-universal hash functions mapping n bits to m bits. Let F be a family of puncturable PRFs mapping n bits to m bits, which exists by Theorem 1.

Consider the family F' defined as follows: The key space for F' consists of a key K for F and a hash function h chosen from $\mathcal{H}$. We define $F'((K, h), x) = F(K, x) \oplus h(x)$.

Observe that if F were a truly random function, then $F(x) \oplus h(x)$ would still be a truly random function for an independent random choice of h (even if h were public). Thus, because F is a PRF, we have that F' is also a PRF. Similarly, because F is puncturable, so is F': one can puncture the key (K, h) on a set S by simply puncturing K on the same set S.

All that remains is to show that F' is statistically injective. Consider $x \neq x' \in \{0,1\}^n$. Fix any K. By pairwise independence, $\Pr_h[h(x) = h(x') \oplus F(K, x) \oplus F(K, x')] = 2^{-m}$. Taking a union bound over all distinct pairs (x, x') gives us that:

$$\Pr_h[\exists x \neq x' : F'((K, h), x) = F'((K, h), x')] \leq 2^{-(m-2n)}$$

Averaging over the choice of K finishes the proof, by the choice of $m \geq 2n + e$. ∎

Finally, we consider PRFs that are also (strong) extractors over their inputs:

DEFINITION 9. *An* extracting *(puncturable) PRF family with error $\epsilon(\cdot)$ for min-entropy $k(\cdot)$ is a family of (puncturable) PRFs F mapping $n(\lambda)$ bits to $m(\lambda)$ bits such that for all λ, if X is any distribution over $n(\lambda)$ bits with min-entropy greater than $k(\lambda)$, then the statistical distance between $(K \leftarrow \mathrm{Key}_F(1^\lambda), F(K, X))$ and $(K \leftarrow \mathrm{Key}_F(1^\lambda), U_{m(\lambda)})$ is at most $\epsilon(\lambda)$, where U_ℓ denotes the uniform distribution over ℓ-bit strings.*

THEOREM 3. *If one-way functions exist, then for all efficiently computable functions $n(\lambda)$, $m(\lambda)$, $k(\lambda)$, and $e(\lambda)$ such that $n(\lambda) \geq k(\lambda) \geq m(\lambda) + 2e(\lambda) + 2$, there exists an extracting puncturable PRF family that maps $n(\lambda)$ bits to $m(\lambda)$ bits with error $2^{-e(\lambda)}$ for min-entropy $k(\lambda)$.*

PROOF. For ease of notation, we suppress the dependence of n, m, k, e on λ. By Theorem 2, let F be a family of puncturable statistically injective PRFs mapping n bits to $2n + e + 1$ bits, with error $2^{-(e+1)}$. Let $\mathcal{H}$ be a family of 2-universal hash functions mapping $2n+e+1$ bits to m bits.

Consider the family F' defined as follows: The key space for F' consists of a key K for F and a hash function h chosen from $\mathcal{H}$. We define $F'((K, h), x) = h(F(K, x))$.

Observe that if F were a truly random function, then $h(F(x))$ would still be a truly random function for an independent random choice of h (even if h were public), by the Leftover Hash Lemma. Thus, because F is a PRF, we have that F' is also a PRF. Similarly, because F is puncturable, so is F': one can puncture the key (K, h) on a set S by simply puncturing K on the same set S.

Now suppose we have any distribution X over n bits with min-entropy at least $k \geq m+2e+2$. Fix any key K such that $F(K, \cdot)$ is injective. Then the Leftover Hash Lemma implies that that the statistical distance between $(h \leftarrow \mathcal{H}, h(F(K, X)))$ and $(h \leftarrow \mathcal{H}, U_m)$ is at most $2^{-(e+1)}$.

Since the probability that K yields a non-injective $F(K, \cdot)$ is also at most $2^{-(e+1)}$, by a union bound, this implies that that statistical distance between $((K, h), F'((K, h), X))$ and $((K, h), U_m)$ is at most 2^{-e}, as desired. ∎

4. OUR DENIABLE ENCRYPTION SYSTEM

Let λ be a security parameter. Below, we assume that the $\mathsf{Encrypt}_{PKE}(\mathrm{PK}, \cdot; \cdot)$ algorithm accepts a one-bit message m and randomness of length $\ell_e \geq 2$, and outputs ciphertexts of length ℓ_c. For ease of notation, we suppress the dependence of these lengths on λ.

Our Encrypt program will take a one-bit message m, and randomness $u = (u[1], u[2])$ of length $\ell_1 + \ell_2$, where $|u[1]| = \ell_1$ and $|u[2]| = \ell_2$. We will set $\ell_1 = 5\lambda + 2\ell_c + \ell_e$, and $\ell_2 = 2\lambda + \ell_c + 1$.

We use a PRG that maps $\{0,1\}^\lambda$ to $\{0,1\}^{2\lambda}$. We also make use of three different PRF variants in our construction:

- An puncturable extracting PRF $F_1(K_1, \cdot)$ that accepts inputs of length $\ell_1 + \ell_2 + 1$, and output strings of length ℓ_e. It is extracting when the input min-entropy is greater than $\ell_e + 2\lambda + 4$, with error less than $2^{-(\lambda+1)}$. Observe that $n + \ell_1 + \ell_2 \geq \ell_e + 2\lambda + 4$, and thus if one-way functions exist, then such puncturable extracting PRFs exist by Theorem 3.
- A puncturable statistically injective PRF $F_2(K_2, \cdot)$ that accepts inputs of length $2\lambda + \ell_c + 1$, and outputs strings of length ℓ_1. Observe that $\ell_1 \geq 2 \cdot (2\lambda + \ell_c + 1) + \lambda$, and thus if one-way functions exist, such puncturable statistically injective PRFs exist by Theorem 2.
- A puncturable PRF $F_3(K_3, \cdot)$ that accepts inputs of length ℓ_1 and outputs strings of length ℓ_2. If one-way functions exist, then such a puncturable PRF exists by Theorem 1.

The public key consist of indistinguishability obfuscation of the two programs described in Figures 1 and 2.

Encrypt

Constants: Public key PK and keys K_1, K_2, and K_3.
Input: Message m, randomness $u = (u[1], u[2])$.

```
1. If $F_3(K_3, u[1]) \oplus u[2] = (m', c', r')$ for (proper length) strings $c', r', m'$,
   and $m' = m$,
   and $u[1] = F_2(K_2, (m', c', r'))$, then output $c = c'$ and end.
2. Else let $x = F_1(K_1, (m, u))$. Output $c = \mathsf{Encrypt}_{PKE}(\mathrm{PK}, m; x)$.
```

Figure 1: Program Encrypt

Explain

Constants: keys K_2 and K_3.
Input: Message m, ciphertext c, randomness $r \in \{0,1\}^\lambda$.

```
1. Set $\alpha = F_2(K_2, (m, c, \mathrm{PRG}(r)))$. Let $\beta = F_3(K_3, \alpha) \oplus (m, c, \mathrm{PRG}(r))$.
   Output $e = (\alpha, \beta)$.
```

Figure 2: Program Explain

The analysis of our construction is deferred to the full version of this paper [22].

5. CORE PRIMITIVES FROM INDISTINGUISHABILITY OBFUSCATION

In this section we show how to build up core cryptographic primitives from Indistinguishable Obfuscation. The primitives we build are: public key encryption, short "hash-and-sign" selectively secure signatures, chosen-ciphertext secure public key encryption, non-Interactive zero knowledge proofs (NIZKs), injective trapdoor functions, and oblivious transfer. We show that each of these can be built from Indistinguishability Obfuscation and (using known implications [17, 19]) standard one way functions, with the exception of trapdoor functions which requires an injective one way function.

Interestingly, the use of punctured programmability emerges as a repeated theme in building each of these. In addition, techniques used to realize one primitive will often have applications in another. For example, our NIZK system uses an obfuscated program that takes as input an instance and witness and outputs a signature on the instance if the witness relation holds. Internally, it leverages the structure of our signature scheme.

For compactness we refer the reader to the literature (e.g. [16]) for the definitions of these core primitives. For each primitive we will present our construction and proof of security. In the proof of security we will give a detailed description of a sequence of hybrid arguments (in the full version of this paper [22]). Once the hybrids are established arguing indistinguishability between them is fairly straightforward and we sketch these arguments.

In these proceedings, we describe only our public-key encryption scheme. For our other results, see the full version of this paper [22].

5.1 Public Key Encryption

We note that the implication that public-key encryption follows from indistinguishability obfuscation and one-way functions already follows from claims in the literature: Indistinguishability obfuscation implies Witness Encryption [12]; Witness Encryption and one-way functions imply public-key encryption [13]. However, that chain of implications, if followed, would result in a public-key encryption scheme where ciphertexts are themselves obfuscated programs. Our approach, on the other hand, yields public-key encryption where public keys are obfuscated programs, and ciphertexts are short.

Let PRG be a pseudo random generator that maps $\{0,1\}^\lambda$ to $\{0,1\}^{2\lambda}$. Let F be a puncturable PRF that takes inputs of 2λ bits and outputs ℓ bits. We describe our encryption algorithm which encrypts for the message space $\mathcal{M} = \{0,1\}^\ell$.

- $\mathsf{Setup}(1^\lambda)$: The setup algorithm first chooses a puncturable PRF key K for F. Next, it creates an obfuscation of the program PKE Encrypt of Figure 3. The size of the program is padded to be the maximum of itself and PKE Encrypt* of Figure 4, which is also included here to give a hint about the proof of security. The public key, PK, is the obfuscated program. The secret key SK is K.

- $\mathsf{Encrypt}(\mathrm{PK}, m \in \mathcal{M})$: The encryption algorithm chooses a random value $r \in \{0,1\}^\lambda$ and runs the obfuscated program of PK on inputs m, r.

- $\mathsf{Decrypt}(\mathrm{SK}, c = (c_1, c_2))$ The decryption algorithm outputs $m' = F(K, c_1) \oplus c_2$.

THEOREM 4. *If our obfuscation scheme is indistinguishably secure,* PRG *is a secure pseudo random generator, and* F *is a secure punctured PRF, then our PKE scheme is IND-CPA secure.*

Acknowledgements

We are grateful to Mihir Bellare for his feedback on earlier versions of this paper. We are indebted to Vanishree Rao for her generous assistance in preparing this proceedings version.

6. REFERENCES

[1] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.

```
                              PKE Encrypt

Constants: Punctured PRF key $K$.
Input: Message $m \in \{0,1\}^\ell$, randomness $r \in \{0,1\}^\lambda$.

  1. Let $t = PRG(r)$.
  2. Output $c = (c_1 = t, c_2 = F(K,t) \oplus m)$.
```

Figure 3: Program PKE Encrypt

```
                              PKE Encrypt*

Constants: Punctured PRF key $K(\{t^*\})$.
Input: Message $m \in \{0,1\}^\ell$, randomness $r \in \{0,1\}^\lambda$.

  1. Let $t = PRG(r)$.
  2. Output $c = (c_1 = t, c_2 = F(K,t) \oplus m)$.
```

Figure 4: Program PKE Encrypt*

[2] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

[3] D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. *IACR Cryptology ePrint Archive*, 2013:352, 2013.

[4] E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. *IACR Cryptology ePrint Archive*, 2013:401, 2013.

[5] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *CRYPTO*, pages 868–886, 2012.

[6] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.

[7] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *CRYPTO*, 2011.

[8] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In *CRYPTO*, pages 90–104, 1997.

[9] W. Diffie and M. E. Hellman. New directions in cryptography, 1976.

[10] M. Dürmuth and D. M. Freeman. Deniable encryption with negligible detection probability: An interactive construction. In *EUROCRYPT*, pages 610–626, 2011.

[11] M. Dürmuth and D. M. Freeman. Deniable encryption with negligible detection probability: An interactive construction. *IACR Cryptology ePrint Archive*, 2011:66, 2011.

[12] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[13] S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters. Attribute-based encryption for circuits from multilinear maps. Cryptology ePrint Archive, Report 2013/128, 2013. http://eprint.iacr.org/.

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

[15] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013.

[16] O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.

[17] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions (extended abstract). In *FOCS*, pages 464–479, 1984.

[18] S. Goldwasser and S. Micali. Probabilistic encryption. *Jour. of Computer and System Science*, 28(2):270–299, 1984.

[19] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[20] A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications. *IACR Cryptology ePrint Archive*, 2013:379, 2013.

[21] T. Moran and A. Rosen. There is no indistinguishability obfuscation in pessiland. *IACR Cryptology ePrint Archive*, 2013:643, 2013.

[22] A. Sahai and B. Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. Cryptology ePrint Archive, Report 2013/454, 2013. http://eprint.iacr.org/.