2013 IEEE 54th Annual Symposium on Foundations of Computer Science

Candidate Indistinguishability Obfuscation and Functional Encryption for all circuits (Extended Abstract)

Sanjam Garg *IBM Research*
Craig Gentry *IBM Research*
Shai Halevi *IBM Research*
Mariana Raykova *IBM Research*
Amit Sahai *UCLA*
Brent Waters *UT Austin*

***Abstract*—In this work, we study *indistinguishability obfuscation* and *functional encryption* for general circuits:**

Indistinguishability obfuscation requires that given any two equivalent circuits C_0 and C_1 of similar size, the obfuscations of C_0 and C_1 should be computationally indistinguishable.

In functional encryption, ciphertexts encrypt inputs x and keys are issued for circuits C. Using the key SK_C to decrypt a ciphertext $\mathrm{CT}_x = \mathsf{Enc}(x)$, yields the value $C(x)$ but does not reveal anything else about x. Furthermore, no collusion of secret key holders should be able to learn anything more than the union of what they can each learn individually.

We give constructions for indistinguishability obfuscation and functional encryption that supports all polynomial-size circuits. We accomplish this goal in three steps: -

- **We describe a candidate construction for indistinguishability obfuscation for $\mathbf{NC}^1$ circuits. The security of this construction is based on a new algebraic hardness assumption. The candidate and assumption use a simplified variant of multilinear maps, which we call *Multilinear Jigsaw Puzzles*.**
- **We show how to use indistinguishability obfuscation for $\mathbf{NC}^1$ together with Fully Homomorphic Encryption (with decryption in $\mathbf{NC}^1$) to achieve indistinguishability obfuscation for all circuits.**
- **Finally, we show how to use indistinguishability obfuscation for circuits, public-key encryption, and non-interactive zero knowledge to achieve functional encryption for all circuits. The functional encryption scheme we construct also enjoys succinct ciphertexts, which enables several other applications.**

The first author was a PhD student at UCLA when this work was done.

The first and fifth authors were supported in part from NSF grants 1228984, 1136174, 1118096, 1065276, 0916574 and 0830803, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. The views expressed are those of the author and do not reflect the official policy or position of the National Science Foundation, or the U.S. Government.

The second and third authors were supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D11PC20202. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

The fourth author is supported by NSF Grant No.1017660.

The sixth author is supported by NSF CNS-0915361 and CNS-0952692, CNS-1228599, DARPA N11AP20006, Google Faculty Research award, the Alfred P. Sloan Fellowship, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

I. Introduction

In this work we study two long-standing open feasibility questions in cryptography: secure program obfuscation, and functional encryption.

Obfuscation. Roughly speaking, program obfuscation aims to make a computer program "unintelligible" while preserving its functionality. The formal study of program obfuscation was initiated by Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang [1], [2]. Unfortunately, they showed that the most natural simulation-based formulation of program obfuscation (a.k.a. "black-box obfuscation") is impossible to achieve for general programs, in a very strong sense: They showed that there exist *unobfuscatable* functions – a family of functions $\{f_s\}$ such that given *any* circuit that implements f_s, an efficient procedure can extract the secret s; however, any efficient adversary given only black-box access to f_s cannot guess even a single bit of s with non-negligible advantage. Indeed, such unobfuscatable function families exist with very low circuit complexity (i.e. in $\mathbf{TC^0}$ under widely believed intractability assumptions), so one cannot hope for general purpose obfuscators achieving a natural simulation-based definition of obfuscation even for very low complexity classes.

Faced with this impossibility result, Barak et al. [1], [2] suggested another notion of program obfuscation called *indistinguishability obfuscation*: An indistinguishability obfuscator $i\mathcal{O}$ for a class of circuits $\mathcal{C}$ guarantees that given two *equivalent* circuits C_1 and C_2 from the class, the two distribution of obfuscations $i\mathcal{O}(C_1)$ and $i\mathcal{O}(C_2)$ should be computationally indistinguishable. We note that if the circuit class $\mathcal{C}$ has efficiently computable canonical forms, then the computation of that canonical form would already be an indistinguishability obfuscator [2], [3]. Since their introduction in 2001, it has been an open problem to construct indistinguishability obfuscators for any natural circuit class that is not known to admit efficiently computable canonical forms. In particular, the central open question regarding indistinguishability obfuscation has been:

Do there exist indistinguishability obfuscators for all polynomial-size circuits?

It is important to note that unlike simulation-based definitions of obfuscation, it is not immediately clear how useful indistinguishability obfuscators would be. Perhaps the strongest philosophical justification for indistinguishability obfuscators comes from the work of Goldwasser and Rothblum [3], who showed that (efficiently computable) indistinguishability obfuscators achieve the notion of *Best-Possible Obfuscation* [3]: Informally, a best-possible obfuscator guarantees that its output hides as much about the input circuit as any circuit of a certain size.

Nevertheless, very few applications of indistinguishability obfuscators were described in the literature, in fact the only application that we are aware of is to removing software watermarks [2]. The main contributions of this work are to **(1)** construct indistinguishability obfuscators for all circuits, and **(2)** show how to use indistinguishability obfuscators to solve the central open problem in the area of Functional Encryption.

Functional Encryption. In Functional Encryption, ciphertexts encrypt inputs x and keys are issued for strings y. The striking feature of this system is that using the key SK_y to decrypt a ciphertext $\mathrm{CT}_x = \mathsf{Enc}(x)$, yields the value $F(x, y)$ but does not reveal anything else about x. Furthermore, *any arbitrary* collusion of key holders relative to many strings y_i does not yield any more information about x beyond what is "naturally revealed" by the functionality (i.e. $F(x, y_i)$ for all i).

Functional Encryption started with the notion of attribute-based encryption (ABE) [4], [5] and evolved over time [6], [7]. The term functional encryption first appeared in [8], and formal definitions and constructions were subsequently given in [9], [10]. Earlier influences include Identity-Based Encryption (IBE) [11], [12], [13] and searching on encrypted data [14], [15].

The general notion of functional encryption subsumes the notion of ABE: In ABE, encrypted messages consist of a secret payload m and a *public* attribute string x, where the secret key SK_y permits the recovery of the payload m if and only if $F(x, y) = 1$. While both ABE and FE require full collusion resilience, the critical difference between ABE and general FE is that ABE does not ensure the secrecy of the attribute string x. Recent breakthrough works [16], [17], [18] show how to achieve ABE for arbitrary circuits F, however these constructions do not provide any secrecy for the x input.

In contrast to ABE, existing FE schemes prior to our work are much more limited in power, with the state of the art roughly limited to the inner-product construction of Katz et al. [7] and its adaptation to lattice-based cryptography by Agrawal et al. [19]. In these systems ciphertexts and keys are associated with vectors $\mathbf{x}, \mathbf{y}$, and on decryption we only learn if the dot product $\mathbf{x} \cdot \mathbf{y}$ is zero. While there are interesting applications of this basic functionality, it is a far cry from the goal of realizing functional encryption for general functions, or even a large general class of circuits. There are also constructions that achieve only limited-collusion notions of security (and also several impossibility results for strong simulation-based security notions). We discuss these in Section X below. The central open question regarding functional encryption has been:

Does there exist a functional encryption scheme supporting all polynomial-size circuits?

II. Our Results

Our first result is a candidate construction of an indistinguishability obfuscator $i\mathcal{O}$ for all polynomial-size, log-depth circuits (i.e. $\mathbf{NC}^1$). The security of our construction relies on a new algebraic hardness assumption. We provide evidence for the plausibility of our assumption by proving it to be true in a specific generic model that seems to capture most "natural attacks" on this type of construction.

Our construction and assumption are staged in a framework that we call *Multilinear Jigsaw Puzzles*. Multilinear Jigsaw Puzzles are a simplified variant of multilinear maps, in which only the party who generated the system parameters can "encode elements in the exponent" of the different groups. These Multilinear Jigsaw Puzzles can be constructed by *eliminating* some components from the GGH (asymmetric) "approximate multilinear maps" [20]. Most notably, we eliminate the elements of the GGH setup that give rise to the most potent cryptanalytic attacks, including the "weak discrete log" attack [20]. We can also instantiate Multilinear Jigsaw Puzzles by eliminating some parts from the recent work of [21] over the integers.

After constructing indistinguishability obfuscators for $\mathbf{NC}^1$ from Multilinear Jigsaw Puzzles, we use these indistinguishability obfuscators in a black-box manner to obtain the following results:

- Using indistinguishability obfuscator for $\mathbf{NC}^1$ together with any (leveled) fully homomorphic encryption (FHE) scheme with decryption in $\mathbf{NC}^1$ (e.g. [22], [23], [24], [25], [26]), we show how to obtain an indistinguishability obfuscator for all polynomial-size circuits. The essential new idea here is to use a "two-key" encryption technique, similar in spirit to Naor-Yung [27].
- Using indistinguishability obfuscator for polynomial-size circuits, together with injective one-way functions, public-key encryption, and a novel variant of Sahai's simulation-sound non-interactive zero knowledge [28] proofs, we show how to obtain functional encryption schemes supporting all polynomial-size circuits.
  Our construction achieves selective indistinguishability security, which can be amplified to full indistinguishability security using complexity leveraging. Finally,

using the recent transformation of De Caro et al [29] one can obtain meaningful simulation-based security for functional encryption for all circuits.

Our construction furthermore achieves *succinct* ciphertexts: The size of the ciphertexts in our scheme does not depend on potential secret key circuit sizes or even depth (nor do they depend on any parameter sizes needed for Multilinear Jigsaw Puzzles).

We emphasize that these results use our indistinguishability obfuscator for $\mathbf{NC}^1$ as a black box, and do not need to make further use of Multilinear Jigsaw Puzzles. We are able to obtain these results through a novel use of indistinguishability obfuscators, by making an intuitive connection between the use of indistinguishability obfuscation and witness indistinguishable proofs. (More on this below.)

III. Multilinear Jigsaw Puzzles

For our construction we use a variant of multilinear maps that offers only a *strict subset* of the functionality, and we term this variant Multilinear Jigsaw Puzzles. These are similar to (the asymmetric version of) the GGH multilinear encoding schemes [20], except that in our setting only the party that generated the system parameters is able to encode elements. Below is an intuitive description of Multilinear Jigsaw Puzzles using multiplicative notation for groups, a more formal treatment can be found in the full-version [30]. Roughly speaking, an instance of a Multilinear Jigsaw Puzzle scheme includes a multilinear map system over groups of prime order p:

$$e : G_1 \times G_2 \times \cdots \times G_k \to G_T.$$

The system parameters allows a user to perform group and multilinear operations, but the parameters need not include canonical generators for the different groups $g_i \in G_i$ and $g_T \in G_T$. A *valid multilinear form* over such system is anything that can be computed using the group operation in the separate groups and the multilinear map. (For example, for $k = 3$ with variables $x_i \in G_1$, $y_i \in G_2$, $z_i \in G_3$ and $w_i \in G_T$, the expression $w_1 \cdot e(x_1, y_3, z_1 z_2)^2 \cdot e(x_2^3 x_4,\ y_1^2, z_2 z_5^3)^5 \cdot w_3^2$ is a valid form.)

In Multilinear Jigsaw Puzzle we view group elements as the puzzle pieces. The intuitive analogy to jigsaw puzzles is that these group elements can only be combined in very structured ways – like jigsaw puzzle pieces, different puzzle pieces either "fit" together or, if they do not "fit", then they cannot be combined in any meaningful way. We view a valid multilinear form in these elements as a suggested solution to this jigsaw puzzle: a valid multilinear form suggests ways to interlock the pieces together. A solution is correct if evaluating the multilinear form on the given elements yields the the unit in the target group $g_T^0 \in G_T$.

Thus, a Multilinear Jigsaw Puzzle scheme consists of just two algorithms, the Jigsaw generator that generates the system parameters and some group elements, and the Jigsaw verifier that checks whether a given solution is correct for these elements. In more detail, the Jigsaw generator outputs some system parameters prms and $k+1$ nonempty sets of elements $S_i = \{x_1^{(i)}, \ldots, x_{n_i}^{(i)}\} \subset G_i$ for $i = 1, \ldots, k, T$. The Jigsaw verifier takes as input $(\mathsf{prms}, S_1, \ldots, S_k, S_T, \Pi)$, where Π and is a valid multilinear form in these elements, and it outputs "yes" if Π evaluates to the unit element in G_T on the given elements and "no" otherwise. Note that the Multilinear Jigsaw Puzzle scheme itself *does not specify which particular multilinear form the Jigsaw Verifier should apply*, this choice will typically come from the application.

The hardness assumptions that we make typically say that two output distributions of the Jigsaw generator are computationally indistinguishable. For instance, an (asymmetric) analog of the BDDH assumption for $k = 2$ could be that $(\{g_1, g_1^a, g_1^b\}, \{g_2, g_2^c\}, \{g_T, g_T^{abc}\})$ is indistinguishable from $(\{g_1, g_1^a, g_1^b\}, \{g_2, g_2^c\}, \{g_T, g_T^r\})$, where g_1, g_2, g_T are generators of G_1, G_2, and G_T, respectively, that satisfy $e(g_1, g_2) = g_T$.

IV. How to Use Indistinguishability Obfuscation

Now that we have constructed an indistinguishability obfuscator, we are faced with the question: what good is an indistinguishability obfuscator? The definition of indistinguishability obfuscation does not make clear what, *if anything*, an indistinguishability obfuscator actually hides about a circuit. In particular, if the circuit being obfuscated was already in an obvious canonical form, then we know that the indistinguishability obfuscator would not need to hide anything. We observe here an analogy to *witness indistinguishable proofs* [31]: if a statement being proven only has a unique witness, then a witness-indistinguishable proof does not need to hide the witness. The way witness indistinguishability can be used is by explicitly constructing statements that can have multiple witnesses. Similarly, we will use indistinguishability obfuscation by constructing circuits that inherently have multiple equivalent forms. We use this analogy to build our main application of functional encryption.

Warmup: Witness Encryption for NP.

As a warmup to see how indistinguishability obfuscation can be applied, we start by considering a cryptographic notion which implicitly *already* considers circuits that inherently have multiple equivalent forms. Recall the notion of Witness Encryption for NP recently introduced in [18]: Given an NP Language L, a witness encryption scheme for L is an encryption scheme that takes as input an instance x and a message bit b, and outputs a ciphertext c. If $x \in L$ and w is a valid witness for x, then a decryptor can use w to decrypt c and recover b. However, if $x \notin L$, then an encryption of 0 should be computationally indistinguishable from an encryption of 1.

We now observe that Indistinguishability Obfuscation for $\mathbf{NC}^1$ implies Witness Encryption for an NP-Complete language such as $L = $SATISFIABILITY. Consider the "point filter function" function [32] $F_{x,b}(w)$, defined as follows: if w is a valid witness for x, then $F_{x,b}(w) = b$. If w is an invalid witness for x, then $F_{x,b}(w) = \perp$. Note that for SATISFIABILITY, it is immediate that $F_{x,b}$ is in NC^1.

Now consider an Indistinguishability Obfuscation of $F_{x,b}$. This obfuscated circuit is a valid witness encryption for x, b. Correctness of decryption is immediate. To see why secrecy holds, note that if $x \notin L$, then both $F_{x,0}$ and $F_{x,1}$ are constant functions that always output $\perp$. Thus by the definition of Indistinguishability Obfuscation, the obfuscations of $F_{x,0}$ and $F_{x,1}$ are computationally indistinguishable.

V. INDISTINGUISHABILITY OBFUSCATION FOR ALL POLYNOMIAL-SIZE CIRCUITS

Next, we show how to use indistinguishability obfuscation for $\mathbf{NC}^1$ and (leveled) fully homomorphic encryption (FHE) with decryption in $\mathbf{NC}^1$ to obtain indistinguishability obfuscation for all polynomial-size circuits. The main idea is to adapt the two-key paradigm [27] to work using indistinguishability obfuscators instead of witness-indistinguishable proofs. This construction is described in the full-version [30].

To obfuscate a circuit C, we choose and publish two FHE keys PK_0 and PK_1. We also give out encryptions of the circuit C under both FHE public keys, yielding ciphertexts c_0 and c_1. Finally, the obfuscation also has the indistinguishability obfuscation of a certain $\mathbf{NC}^1$ circuit C_{Dec0} that we will describe below.

The evaluator, who holds an input x, is told to use the FHE evaluation algorithm with x and *both* ciphertexts c_0 and c_1 to yield encryptions of $C(x)$ under both PK_0 and PK_1: let us call these encryptions e_0 and e_1 respectively. The evaluator also keeps track of all the intermediate bit values encountered during evaluation, which can be seen as a "proof" π that it used x to perform the evaluation correctly on both c_0 and c_1. The evaluator then feeds e_0, e_1, x, π into the obfuscated circuit $i\mathcal{O}(C_{Dec0})$. This circuit C_{Dec1} first checks the proof π to make sure that e_0 and e_1 were correctly computed – note that this can be done easily in $\mathbf{NC}^1$ since the evaluator has included all intermediate bit values encountered during evaluation, so the circuit merely needs to check that each appropriate triple of intermediate bit values respects the AND, OR, or NAND operations that were used during the evaluation process. If the proof checks out, then the circuit decrypts e_0 using SK_0, and outputs this decryption, which should be $C(x)$.

The key insight is that there is another $\mathbf{NC}^1$ circuit C_{Dec1} that is equivalent to C_{Dec0}, which simply decrypts e_1 using SK_1 instead. Because of the proof π that must be provided, both of these circuits always behave identically on all inputs. Furthermore, when using C_{Dec0}, we note that SK_1 is never used anywhere, and therefore the semantic security of the FHE scheme using PK_1 is maintained even given C_{Dec0}. Thus, by alternatively applying the semantic security of the FHE scheme and switching back and forth between C_{Dec0} and C_{Dec1} using the indistinguishability obfuscation property, we can prove that the obfuscation of any two equivalent circuits C and C' are computationally indistinguishable.

VI. FUNCTIONAL ENCRYPTION FOR ALL CIRCUITS

We next sketch our main application of building functional encryption for all circuits. This construction is described in the full-version [30]. The starting point idea for our solution is to pick a standard public-key encryption key pair $(\mathrm{PK}, \mathrm{SK})$ in the setup phase of the Functional Encryption. An encryption of a value x will simply be an encryption of x using the public key PK. The secret key sk_C corresponding to a circuit C be an obfuscation of a program that uses SK to decrypt x, then computes and output $C(x)$. While this solution would work if our obfuscator achieved the black-box obfuscation definition, there is no reason to believe that an indistinguishability obfuscator would necessarily hide SK.

Recall that the indistinguishability security definition for functional encryption requires the adversary to declare two inputs x_0 and x_1, with the promise that all secret keys SK_C that she will ask for will satisfy $C(x_0) = C(x_1)$. Then the security definition requires that the adversary will not be able to distinguish an encryption of x_0 from an encryption of x_1.

A natural first step would be to have two public keys PK_0 and PK_1, and require the encryption of x to consist of encryptions of x under both keys. However, in this case, unlike above, the receiver cannot generate a proof on his own that both ciphertexts encrypt the same message to provide to the obfuscated decryption circuit. Thus, we must require the encryptor to generate such a proof, which must hide x. A natural solution is to have the encryptor generate a non-interactive zero-knowledge (NIZK) proof that the ciphertexts both encrypt the same input. The obfuscated circuit would first check this proof, and if it checks out, it would use one of the two secret keys SK_0 or SK_1 for decryption and evaluation, confident that the output would be the same no matter which secret key it uses. When we are proving that the adversary cannot distinguish an encryption of x_0 from an encryption of x_1, we could move to an intermediate hybrid experiment where the NIZK proof is simulated, and the two ciphertexts are actually to x_0 and x_1 respectively.

Here we encounter the key new problem that we must tackle: when simulating a NIZK proof, we need to change the common reference string, and in all known NIZK systems, with respect to a simulated common reference string, valid proofs of *false* statements exist. Even with the notion of simulation soundness [28], which was devised precisely to deal with this problem, valid proofs of false statements exist, even if they cannot be found by efficient adversaries.

However, the notion of indistinguishability obfuscation requires that circuits be equivalent for *all inputs*, even inputs to the circuit that contain valid proofs for false statements. We address this problem by introducing the notion of *statistically simulation sound* NIZK, which requires that *except with respect to one particular statement to be simulated*, all other valid proofs that exist must be only for true statements[1]. Statistical simulation soundness can be achieved in a standard way using witness indistinguishable proofs. Once we have it, then a similar two-key alternation argument to the one given above allows us to establish security of our functional encryption scheme. We note that full collusion-resistance follows from a standard hybrid argument using the indistinguishability obfuscation definition for each secret key given out, one at a time.

Once we achieve indistinguishability security for functional encryption, this can be upgraded to natural simulation-based definitions of security using the work of De Caro et al. [29]. Furthermore, we note that our construction enjoys the property that ciphertexts are succinct; indeed their size depends only on the public-key encryption scheme and NIZK being used, and does not depend on the underlying details of the obfuscation mechanism in any way (and therefore it does not depend on the parameters needed for our Multilinear Jigsaw Puzzles). We also place no a-priori bound on the circuit size of the circuits used for secret key generation.

A. Ciphertext Succinctness and Applications

One thing we emphasize is that ciphertexts in our scheme are very succinct in that their size depends only upon the message size and security parameter. In fact, if the right combination of public key encryption and NIZK is used the ciphertext size and encryption time can be considered small in a practical sense. Prior solutions for (the weaker) primitives of ABE for circuits [16], [17] and single use functional encryption [33] achieved ciphertexts sizes and encryption times that were proportional to the maximum depth of the function to be evaluated. (It should be noted that these solutions rely on weaker assumptions.)

Following Parno et. al. [34] we get a delegatable computation scheme with where the client's work is independent of the circuit size. Following Goldwasser et. al. [33] we get a reusable Yao garbled circuits where the work to reuse a garbled circuit is proportional to the input size and independent of the size and depth of the circuit.

Finally, we get an efficient Private Linear Broadcast Encryption (PLBE) scheme as described by Boneh, Sahai, and Waters [35], which they show implies a secure traitor tracing [36] system. This final application requires collusion resistance.

[1]Because this statement must be fixed in advance, the resulting security proof is for selective security. Selective security can be "upgraded" to full security using a standard complexity leveraging argument, by assuming subexponential security of the underlying cryptographic primitives that we use.

VII. An application to Restricted-Use Software

We have already given evidence that indistinguishability obfuscation has significant applications to cryptography. Here, we discuss an application of indistinguishability obfuscation to a question that does not deal with encryption or authentication, but rather *restricted-use software:*

Software developers will often want to release a demo or restricted use version of their software that limits the features that are available in a full version. In some cases a commercial software developer will do this to demonstrate their product; in other cases the developer will want to make multiple tiers of a product with different price points. In other domains, the software might be given to a partner that is only partially trusted and the developer only wants to release the features needed for the task.

Ideally, a developer could create a downgraded version of software simply by starting with the full version and then turning off certain features at the interface level — requiring minimal additional effort. However, if this is all that is done, it could be easy for an attacker to bypass these controls and gain access to the full version or the code behind it. The other alternative is for a software development team to carefully excise all unused functionality from the core of the software. Removing functionality can become a very time consuming task that could itself lead to the introduction of software bugs. In addition, in many applications it might be unclear what can and cannot remain for a restricted use version.

One immediate solution is for a developer to restrict the use at the interface level and then release an obfuscated version of the program. For this application indistinguishability obfuscation suffices, since by definition a version restricted in the interface is indistinguishable from an obfuscated program with equivalent behavior that has its smarts removed at the start.

VIII. Stronger notions of obfuscation

In this work, we give new constructions for secure obfuscation. While indistinguishability obfuscation is the technical focus of our work, we note that we know of no actual attacks on our obfuscation scheme as a black-box obfuscator *except for* the attacks that arise on specific circuit classes that are known to be unobfuscatable. This leads to the intriguing possibility that our construction (or future alternative constructions) could achieve black-box obfuscation for a large class of circuits, that somehow exclude the problematic examples that have been identified. Indeed, as observed by Goldwasser and Rothblum [3], indistinguishability obfuscation *must* yield virtual black-box obfuscation *if* such a virtual black-box obfuscation is possible for the function being obfuscated. In particular, while much study still needs to be done, we currently have no reason not to conjecture that our obfuscation mechanism suffices for such applications as:

- **Secure Patching of Software:** If a new malware vulnerability is found in software, there is a risk that releasing a software patch will allow attackers to become aware of a vulnerability before the patch has a chance to fully circulate among users. Obfuscation offers a solution concept: an initial patch can be released in obfuscated form, and then transitioned to a more efficient un-obfuscated patch once large-scale adoption has occurred for the initial patch. Here, the assumption would be that the obfuscated patch would hide where the vulnerability in the software was (at least as well as the original vulnerable software did).
- **"Intellectual Property" Protection:** If a new algorithm is developed for solving a problem, and there is a worry that available legal protections will not suffice for preventing reverse-engineering the algorithm, then obfuscation offers a natural solution: Here, the general required security property seems very close to black-box obfuscation, but if only certain parts of the algorithm is novel, weaker yet sufficient obfuscation security guarantees may be formalizable in some contexts.

IX. Future Directions in Obfuscation

Our work opens up several new possibilities in obfuscation, below we discuss some of them.

- **The Power of Indistinguishability Obfuscation:** While a black box obfuscator immediately results in several turnkey applications, our work shows that indistinguishability obfuscation combined with adroit use of other cryptographic primitives can give rise to powerful functionalities. Seeing that indistinguishability obfuscation can be much more powerful than it initially appears, an important line of research is to see how far we can push the limits of what it gives us.
- **Indistinguishability Obfuscation from Weaker Assumptions:** The argument of security candidate for Indistinguishability Obfuscation clearly has a significant heuristic component. In future research, we can hope to gain better understanding of these heuristics and eventually achieve constructions under more standard assumptions. A major open problem is to obtain an unconditional proof for our scheme (or a variant of it) in a less idealized generic model, such as the (plain) generic multilinear map model, as opposed to the generic colored matrix model that we use. The eventual goal of this line of research may be to get an Indistinguishability Obfuscator provably secure under the Learning with Error (LWE) assumption (with the aid of complexity leveraging), perhaps using techniques similar to those developed in the context of FHE.
- **Studying the Possibility of Black Box Obfuscation:** As mentioned earlier we know of no actual attacks on our obfuscation scheme as a black-box obfuscator except for attacks that arise on specific circuit classes that are known to be unobfuscatable. A clear line of research is to better understand the potential for our construction to serve as a black box obfuscator, by identifying amenable function classes. One thrust is cryptanalysis, namely finding non-generic attacks when using our scheme, perhaps even against "natural" functionalities. A second direction is to attempt to gain some evidence of the viability for black box security of our construction in some heuristic model such as the generic group model. Of course, such a study would not be limited to our construction and we expect that other interesting candidates or variants of our construction would emerge in the near future.
- **Improving the Practical Efficiency:** While our current obfuscation construction runs in polynomial-time, it is likely too inefficient for most practical problems. An important objective is to improve the efficiency for use in practical applications.
- **Reasoning about Obfuscation of Learnable Programs in Practice:** A final important task is to begin to close the gap between the theoretical definitions for obfuscation and informal expectations that often arise in practice. Large software development projects can consume millions of dollars and hundreds of person years. A company that invested such an effort might reasonably expect that a "good" obfuscator would provide some real and tangible protection. For example, the company might hope that releasing an obfuscated version of their software would not be any "worse than" providing remote access to a server running the software.
  One issue is that despite the substantial investment, the obfuscated program may be learnable in that its behavior can be completely determined by a (large) polynomial number of input queries. If this is the case, then simply giving out the program in the clear would technically count as valid obfuscation from the cryptographic definition even though this strategy would be very disappointing in a real sense.
  We would like to be able to bridge this gap between theory and practice and figure out ways to rigorously define and argue about meaningful obfuscation notions for learnable programs. Getting a better understanding of this gap appears critical for using obfuscation in non-cryptographic applications. We emphasize that an important initial step is to simply understand this problem better.

X. Other Related Work

Some recent work has focused on a collusion-bounded form of functional encryption [37], [38], [33], where security is only ensured so long as an attacker does not acquire more than some predetermined number of keys. This concept is very similar to the manner in which one-time signatures relax the general notion of signatures. In these settings, collusion-bounded FE has been achieved for general circuits [37], [38], and in the case of single-key FE this has been done with quite succinct ciphertexts [33]. Certain collusion-bounded functional encryption schemes with special properties suffice for several interesting applications such as delegated computation and most notably reusable garbled circuits [33]. However, in the general use scenario envisioned for FE [9], unbounded collusion resistance is essential, as a single user or a collection of users would be holding multiple secret keys. This is the focus of our work.

Recent work demonstrated that certain strong simulation-based formulations of the goal of functional encryption are impossible to meet [9], [39]. Our work focuses on realizing the indistinguishability based definition of FE, however recent work has shows that indistinguishability security for general circuits can be used to achieve meaningful simulation-based security as well [29].

XI. Intuition: Indistinguishability Obfuscation Candidate for $\mathbf{NC}^1$

To help explain the role of the different components in our cryptosystem, we describe below a trajectory that begins with Kilian's protocol for oblivious $\mathbf{NC}^1$ circuit evaluation, and modifying it until we end up with our obfuscation candidate for $\mathbf{NC}^1$ circuits.

Starting Point: Barrington and Kilian. Barrington's theorem [40] that bounded-width branching programs can compute $\mathbf{NC}^1$ circuits played a major role in cryptography [41], [42], [43], providing a simple algebraic structure that facilitates structure-preserving randomization (as well as simple and natural defenses to standard attacks). In a nutshell, Barrington showed how to transform an arbitrary fan-in-2, depth-d boolean circuit $C : \{0,1\}^\ell \to \{0,1\}$ into a "width-5 permutation branching program" BP of length $n \leq 4^d$. This program is defined by a sequence $BP = \{(\mathsf{inp}(i), A_{i,0}, A_{i,1})\}_{i=1}^n$, where the function $\mathsf{inp} : [n] \to [\ell]$ describes what input bit is examined in the i'th step, and the $A_{i,b}$'s are permutations in S_5 (which we can view as 5×5 permutation matrices). On any particular ℓ-bit input $(b_1 b_2 \ldots b_\ell)$, the evaluation of BP consists of computing the product matrix $P = \prod_{i=1}^n A_{i,b_{\mathsf{inp}(i)}}$, outputting one if P is the identity and zero otherwise.

Let us recall Kilian's protocol [42] in which the two players, Alice and Bob, evaluate an $\mathbf{NC}^1$ circuit on their joint input. Let $C(x, y)$ denote that circuit, with Alice's input x and Bob's input y, and $|x| + |y| = \ell$. Invoking Barrington's theorem, we get a length-n branching program BP that computes the same function as C. Alice begins the protocol by choosing n random invertible matrices $\{R_i\}_{i=1}^n$ over $\mathcal{Z}_p$, computing their inverses, and then setting $\tilde{A}_{i,b} = R_{i-1} A_{i,b} R_i^{-1}$ for all $i \in [n], b \in \{0,1\}$ (with index arithmetic modulo n).[2] Call the new program with $\tilde{A}_{i,b}$ the *randomized branching program* RBP. It is clear that RBP evaluates the same function as the original BP. Denoting the joint ℓ-bit input by $\chi = (x|y)$, Alice sends to Bob the matrices corresponding to her input x, $\{\tilde{A}_{i,\chi_{\mathsf{inp}(i)}} : i \in [n], \mathsf{inp}(i) \leq |x|\}$. Next they use oblivious transfer protocol by which Bob learns the matrices corresponding to his input y, $\{\tilde{A}_{i,\chi_{\mathsf{inp}(i)}} : i \in [n], \mathsf{inp}(i) > |x|\}$. Now Bob can put the matrices in the proper order to compute the product $P = \prod_{i \in [n]} \tilde{A}_{i,\chi_{\mathsf{inp}(i)}}$, and hence $C(x, y)$. Alice learns nothing about Bob's input, and Kilian shows that Bob does not learn anything about Alice's input since the randomization hides Alice's input perfectly.

Scheme number one: Kilian's protocol as-is. Starting our journey from Kilian to $i\mathcal{O}$, we think of Alice as the obfuscator and Bob as the evaluator. The randomized branching program RBP for a universal circuit $C(x, y)$ in $\mathbf{NC}^1$ is generated during system startup. The x input corresponds to the specific circuit being obfuscated and y the input on which it is evaluated. The obfuscation consists of the matrices corresponding to her input x, $\{\tilde{A}_{i,\chi_{\mathsf{inp}(i)}} : i \in [n], \mathsf{inp}(i) \leq |x|\}$ and all the matrixes for the inputs corresponding to y, namely $\{\tilde{A}_{i,b} : i \in [n], b \in 0,1, \mathsf{inp}(i) > |x|\}$. Bob during evaluation chooses the matrices corresponding to his input y, putting together with the matrices corresponding to input x, evaluates the program outputting $C(x, y)$.

Unsurprisingly, this scheme is broken. Informally speaking, however, we can prove that all attacks on this basic scheme fall into three categories:

- **Partial Evaluation Attacks.** A partial evaluation attack occurs when the adversary computes "partial" products $\left(\tilde{A}_{i,\chi_{\mathsf{inp}(i)}} \cdot \tilde{A}_{i+1,\chi_{\mathsf{inp}(i+1)}} \cdots \tilde{A}_{j,\chi_{\mathsf{inp}(j)}}\right)$ and $\left(\tilde{A}_{i,\chi'_{\mathsf{inp}(i)}} \cdot \tilde{A}_{i+1,\chi'_{\mathsf{inp}(i+1)}} \cdots \tilde{A}_{j,\chi'_{\mathsf{inp}(j)}}\right)$, corresponding to two inputs $\chi = (x, y)$ and $\chi' = (x, y')$, and compares these against each other. Note that the outer randomization matrices R_{i-1} and R_j^{-1} will be the same. This would allow the adversary to learn whether two partial computations in the branching program yield the same value, revealing information about the input x.
- **Mixed Input Attacks.** A mixed input attack occurs when the adversary performs the matrix product involved in a correct computation, but doesn't respect the inp function. That is, for different steps in the BP that

[2]Kilian uses random permutation matrices for the R_i's, but that aspect of the protocol is not relevant to our discussion.

consider the same input bit y_i, it uses some matrices that correspond to $y_i = 0$ and others that correspond to $y_i = 1$. Again, this could yield information about x beyond just the final output $C(x, y)$.

- **Attacks that do not respect algebraic structure or are not multilinear.** These attacks must either fail to respect the algebraic structure of the matrices over $\mathcal{Z}_p$, or compute non-multilinear algebraic functions over these matrices[3]. Informally speaking, we prove (in a particular idealized generic model), that all multilinear attacks over matrices must fall into one of the two attack categories above.

Dealing with "other attacks" using Multilinear Jigsaw Puzzles. We begin by addressing the possibility of non-multilinear attacks, i.e., attacks that do things other than evaluating multilinear forms on the given elements. This is precisely where Multilinear Jigsaw Puzzles come in. In our case the Jigsaw Generator will be the obfuscator, the Jigsaw Specifier will give us the plaintext matrices as above, and the Jigsaw Verifier will be used when evaluating the program in order to decide if the resulting matrix is the identity or not. Informally speaking, the Jigsaw Generator will generate a puzzle system with multi-linearity n, where n is the length of the branching program to be obfuscated. Instead of revealing the matrices $\tilde{A}_{i,b}$ in cleartext, the obfuscator encodes each entry of the ith matrices $\tilde{A}_{i,b}$ relative to the singleton index set $\{i\} \subseteq [n]$.[4] Intuitively, this encoding makes it difficult to mount attacks that are not multilinear in the input elements. Roughly speaking, our hardness assumption states that it is infeasible to distinguish the encoding of two equal-length sequences of matrices that have the same behavior in terms of their products.

Dealing with Partial Evaluation Attacks: "Bookends" and Higher Dimensional Matrices. Note that encoding the matrices in the Multilinear Jigsaw Puzzle framework does nothing to prevent Partial Evaluation Attack and Mixed Input Attacks, as those attacks arise even when the adversary respects the matrix product structure.

To address the partial evaluation attack, we would like to add special "bookend" components at the start and end of the computation, without which no partial computation can be compared to any other. To do this, first we embed the matrices $A_{i,b}$ inside higher-dimension matrices of the form

$$D_{i,b} \sim \begin{pmatrix} \$ & & & \\ & \ddots & & \\ & & \$ & \\ & & & A_{i,b} \end{pmatrix},$$

[3]For example, Ishai [44] pointed out to us that computing matrix inverses, a function of algebraic degree -1, can lead to some non-trivial attacks on this scheme.

[4]In group notations, this is the equivalent of encoding the entries of that matrix in the exponent of the i'th group G_i.

$$\tilde{D}_{i,b} \sim R_{i-1} \times D_{i,b} \times R_i^{-1},$$

where the \$'s are random coefficients on main diagonal that are *unique* to each matrix, the unspecified entries are 0, and the R_i's are random matrices over Z_p (which will be invertible with overwhelming probability).

Intuitively, encoding the matrices $\tilde{D}_{i,b}$ relative to the singleton insex set $\{i\}$ allows us to evaluate the multilinear form corresponding to their product, thereby recovering an encoding of the matrix $R_0 \cdot P \cdot R_n^{-1}$ (with P the product matrix) relative to the index set $[n]$. But now P is a high-dimension matrix with random random entries along the main diagonal, and the output of the function is encoded in the P's lower-right 5×5 quadrant (depending on whether or not it is the identity matrix). How can we "decode" the final output? To accomplish this, we add the special "bookend" terms that exactly cancel out the random diagonal entries.

Specifically, let the random part of the main diagonal have dimension $2m$ (for m to be specified shortly), so the matrices all have dimension $2m+5$. We generate two $(2m+5)$-vectors $\mathbf{s}$ and $\mathbf{t}$ of the form:

$$\mathbf{s} \sim (0 \ldots 0 \; \$ \ldots \$ \;\; \text{-}\; \mathbf{s}^* \text{-}\;), \qquad \mathbf{t} \sim (\$ \ldots \$ \;\; 0 \ldots 0 \;\; \text{-}\; \mathbf{t}^* \text{-}\;),$$

where the first 2 blocks of each vector have size m, and the last blocks have size 5. We also compute $\tilde{\mathbf{s}} = \mathbf{s}R_0^{-1}$ and $\tilde{\mathbf{t}} = R_n\mathbf{t}$.

Increasing the multilinearity level from n to $n+2$, the obfuscator adds encodings of the two bookend vectors $\hat{\mathbf{s}} = \mathsf{Enc}_{\{1\}}(\tilde{\mathbf{s}})$ and $\hat{\mathbf{t}} = \mathsf{Enc}_{\{n+2\}}(\tilde{\mathbf{t}})$ (where $\mathsf{Enc}_{\{i\}}(\cdot)$ denotes an encoding relative to the singleton $\{i\}$), and also adds to the obfuscation an encoding of the scalar $p^* = \langle \mathbf{s}^*, \mathbf{t}^* \rangle$ at level $[n+2]$. Now the evaluator can compute a level-$[n+2]$ encoding of

$$p = \tilde{\mathbf{s}} \cdot (R_0 P R_n^{-1}) \cdot \tilde{\mathbf{t}}^T = \mathbf{s}^* \cdot A \cdot \mathbf{t}^{*T} \quad \text{and} \quad p' = p - p^*$$

where the first $2m$ dimensions disappear due to the special structure of the $\mathbf{s}$, $\mathbf{t}$ and P. If A is indeed the identity matrix, then $\mathbf{s}^* \cdot A \cdot \mathbf{t}^{*T} = \langle \mathbf{s}^*, \mathbf{t}^* \rangle = p^*$, and so $p' = 0$ which can be verified using the Jigsaw Verifier. If A is some other permutation matrix, equality will not hold with high probability.

Dealing with Mixed Input Attacks: Multiplicative Bundling. Finally, we must add a method that prevents the adversary from computing "mixed up" products of some matrices that correspond to $y_i = 0$ and other that correspond to some $y_i = 1$ (for the same input bit y_i). To defend against this threat we use a *multiplicative bundling* technique. The idea is simple, to prevent mixing and matching of components, Alice adds additional scalar multiples that help ensure that every "non-faithful" execution leads to a "garbled" output. Namely we choose scalars $\{\alpha_{i,b} : i \in [n], b \in \{0, 1\}\}$

at random and set

$$D_{i,b} \sim \begin{pmatrix} \$ & & & \\ & \ddots & & \\ & & \$ & \\ & & & \alpha_{i,b} \cdot A_{i,b} \end{pmatrix},$$

$$\tilde{D}_{i,b} \sim R_{i-1} \times D_{i,b} \times R_i^{-1}.$$

Unfortunately with these multiplicative factors, the encoded value of p^* is no longer sufficient to decode the output. To fix this issue, we need to provide in the public parameters something else that helps us to "cancel out" the $\alpha_{i,b}$'s. For that purpose, we introduce a "parallel" dummy branching program that computes the constant 1 function into our obfuscated program, and decoding the result is done by comparing the results from both programs.

Specifically, at system setup time we also generate a second set of matrices R'_i, vectors $\mathbf{s}'$ and $\mathbf{t}'$ with the same form as $\mathbf{s}$ and $\mathbf{t}$ where $\langle \mathbf{s}', \mathbf{t}' \rangle = \langle \mathbf{s}, \mathbf{t} \rangle$, and choose scalars $\{\alpha'_{i,b} : i \in [n], b \in \{0,1\}\}$ whose product match that of the α's, namely

$$\prod_{\mathsf{inp}(i)=j} \alpha_{i,b} = \prod_{\mathsf{inp}(i)=j} \alpha'_{i,b} \text{ for all } j \in [\ell],\ b \in \{0,1\}.$$

Then we set $\tilde{\mathbf{s}}' = \mathbf{s}' \cdot (R'_0)^{-1}$, $\tilde{\mathbf{t}}' = R'_n \cdot \mathbf{t}'^T$, and

$$D'_{i,b} \sim \begin{pmatrix} \$ & & & \\ & \ddots & & \\ & & \$ & \\ & & & \alpha'_{i,b} \cdot I \end{pmatrix},$$

$$\tilde{D}'_{i,b} \sim R'_{i-1} \times D_{i,b} \times R'^{-1}_i,$$

Note that this has almost exactly the same structure as the "primary" branching program, except all the $A_{i,b}$'s are replaced by the identity matrix (hence this dummy program computes the constant 1 function). Also, this new copy is nearly independent of the primary program, except for the choice of the $\alpha_{i,b}$'s and $\alpha'_{i,b}$'s. The obfuscated program includes the "primary" vectors $\tilde{\mathbf{s}}, \tilde{\mathbf{t}}$ and matrices $\tilde{D}_{i,b}$, as well as the "dummy" vectors $\tilde{\mathbf{s}}', \tilde{\mathbf{t}}'$ and matrices $\tilde{D}'_{i,b}$.

Given all the encoded matrices and vectors, we can decode the output by checking if the two programs compute the same thing. Namely, we compute the difference between the two evaluations

$$\delta \ = \ \hat{\mathbf{s}} \cdot \left(\prod_{i \in [n]} \hat{D}_{i,b_i} \right) \cdot \hat{\mathbf{t}}^T \ - \ \hat{\mathbf{s}}' \cdot \left(\prod_{i \in [n]} \hat{D}'_{i,b_i} \right) \cdot \hat{\mathbf{t}}'^T,$$

and use the Jigsaw verifier to test if $\delta = 0$.

Observe that these random scalars indeed thwart mixed-evaluation attacks: For every input bit y_j, if we choose all the $\alpha_{i,0}$'s and all the $\alpha'_{i,0}$'s for $\mathsf{inp}(i) = j$ then the product matches and we could get $\delta = 0$ (similarly if we choose all the $\alpha_{i,1}$'s and $\alpha'_{i,1}$'s). However any attempt to mix-and-match these matrices will result in two different random products for the primary and dummy programs, which an adversary can only "cancel out" with probability $1/p$.

Additional Safeguards. In the construction above we choose to set the number m of "random dimensions" that are added to the matrices to more than the multi-linearity level of the system, namely set $m = 2n + 5$. We note that we do not have any concrete reason why even setting $m = 1$ is insecure, but we increase m as an additional safeguard against unanticipated attacks. By doing so, we increase the level of randomness present in both the $D_{i,b}$ matrices, and the individual R_i matrices to be well beyond the maximum multilinearity supported by the Multilinear Jigsaw Puzzle. In particular, this means that written as formal polynomials, each entry of R_i^{-1} is a polynomial with a degree exceeding the maximum multilinearity of the system. Intuitively, this gives us more confidence in our assumption because it seems quite implausible that an adversary that is essentially limited to computations that are "close" to multilinear forms would be able to detangle such high degree computations involving so many degrees of freedom in terms of the randomness used. However, again, we are not aware of any attacks on our scheme even without this additional safeguard.

ACKNOWLEDGMENTS

We are grateful to Boaz Barak, Dan Boneh, Ran Canetti, Angelo De Caro, Shafi Goldwasser, Vipul Goyal, Hemanta Maji, Yuval Ishai, Yael Kalai, Allison Lewko, Omkant Pandey, Omer Paneth, Manoj Prabhakaran, and David Wagner for many illuminating conversations related to obfuscation and functional encryption.

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*, 2001, pp. 1–18.

[2] ——, "On the (im)possibility of obfuscating programs," *J. ACM*, vol. 59, no. 2, p. 6, 2012.

[3] S. Goldwasser and G. N. Rothblum, "On best-possible obfuscation," in *TCC*, 2007, pp. 194–213.

[4] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *EUROCRYPT*, 2005, pp. 457–473.

[5] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *CCS*, 2006.

[6] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *TCC*, 2007, pp. 535–554.

[7] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *EUROCRYPT*, 2008.

[8] A. Sahai and B. Waters, "Slides on functional encryption," PowerPoint presentation, 2008, http://www.cs.utexas.edu/ bwaters/presentations/files/functional.ppt .

[9] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: definitions and challenges," in *TCC*, 2011, pp. 253–273.

[10] A. O'Neill, "Definitional issues in functional encryption," Cryptology ePrint Archive, Report 2010/556, 2010.

[11] A. Shamir, "Identity-based cryptosystems and signature schemes," in *CRYPTO*, 1984, pp. 47–53.

[12] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *CRYPTO*, 2001.

[13] C. Cocks, "An identity based encryption scheme based on quadratic residues." in *IMA Int. Conf.*, 2001, pp. 360–363.

[14] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Security and Privacy*, 2000, pp. 44–55.

[15] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *EUROCRYPT*, 2004, pp. 506–522.

[16] S. Gorbunov, V. Vaikuntanathan, and H. Wee, "Attribute-based encryption for circuits," in *STOC*, 2013.

[17] S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters, "Attribute-based encryption for circuits from multilinear maps," in *CRYPTO*, 2013.

[18] S. Garg, C. Gentry, A. Sahai, and B. Waters, "Witness encryption and its applications," in *STOC*, 2013.

[19] S. Agrawal, D. M. Freeman, and V. Vaikuntanathan, "Functional encryption for inner product predicates from learning with errors," in *ASIACRYPT*, 2011.

[20] S. Garg, C. Gentry, and S. Halevi, "Candidate multilinear maps from ideal lattices," in *EUROCRYPT*, 2013.

[21] J.-S. Coron, T. Lepoint, and M. Tibouchi, "Practical multilinear maps over the integers," in *CRYPTO*, 2013.

[22] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC*, 2009, pp. 169–178.

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

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

[25] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," in *CRYPTO*, 2012, pp. 868–886.

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

[27] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attacks," in *STOC*, 1990, pp. 427–437.

[28] A. Sahai, "Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security," in *FOCS*, 1999, pp. 543–553.

[29] A. D. Caro, V. Iovino, A. Jain, A. O'Neill, O. Paneth, and G. Persiano, "On the achievability of simulation-based security for functional encryption," in *CRYPTO*, 2013.

[30] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Definitional issues in functional encryption," Cryptology ePrint Archive, Report 2013, 2013.

[31] U. Feige and A. Shamir, "Witness indistinguishable and witness hiding protocols," in *STOC*, 1990, pp. 416–426.

[32] S. Goldwasser and Y. T. Kalai, "On the impossibility of obfuscation with auxiliary input," in *FOCS*, 2005, pp. 553–562.

[33] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich, "Succinct functional encryption and applications: Reusable garbled circuits and beyond," in *STOC*, 2013.

[34] B. Parno, M. Raykova, and V. Vaikuntanathan, "How to delegate and verify in public: Verifiable computation from attribute-based encryption," in *TCC*, 2012, pp. 422–439.

[35] D. Boneh, A. Sahai, and B. Waters, "Fully collusion resistant traitor tracing with short ciphertexts and private keys," in *EUROCRYPT*, 2006, pp. 573–592.

[36] B. Chor, A. Fiat, and M. Naor, "Tracing traitors," in *CRYPTO*, 1994, pp. 257–270.

[37] A. Sahai and H. Seyalioglu, "Worry-free encryption: functional encryption with public keys," in *CCS*, 2010, pp. 463–472.

[38] S. Gorbunov, V. Vaikuntanathan, and H. Wee, "Functional encryption with bounded collusions via multi-party computation," in *CRYPTO*, 2012.

[39] S. Agrawal, S. Gorbunov, V. Vaikuntanathan, and H. Wee, "Functional encryption: New perspectives and lower bounds," in *CRYPTO*, 2013.

[40] D. A. Barrington, "Bounded-width polynomial-size branching programs recognize exactly those languages in nc_1," in *STOC*, 1986.

[41] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or a completeness theorem for protocols with honest majority," in *STOC*, 1987, pp. 218–229.

[42] J. Kilian, "Founding cryptography on oblivious transfer," in *STOC*, J. Simon, Ed. ACM, 1988, pp. 20–31.

[43] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway, "Everything provable is provable in zero-knowledge," in *CRYPTO*, 1988, pp. 37–56.

[44] Y. Ishai, "Personal communication," 2013.