# Lockable Obfuscation

Rishab Goyal
UT Austin
rgoyal@cs.utexas.edu

Venkata Koppula
UT Austin
kvenkata@cs.utexas.edu

Brent Waters
UT Austin
bwaters@cs.utexas.edu*

**Abstract**

In this paper we introduce the notion of lockable obfuscation. In a lockable obfuscation scheme there exists an obfuscation algorithm $\mathsf{Obf}$ that takes as input a security parameter $\lambda$, a program $P$, a message $\mathsf{msg}$ and "lock value" $\alpha$ and outputs an obfuscated program $\widetilde{P}$. One can evaluate the obfuscated program $\widetilde{P}$ on any input $x$ where the output of evaluation is the message $\mathsf{msg}$ if $P(x) = \alpha$ and otherwise receives a rejecting symbol $\perp$.

We proceed to provide a construction of lockable obfuscation and prove it secure under the Learning with Errors (LWE) assumption. Notably, our proof only requires LWE with polynomial hardness and does not require complexity leveraging.

We follow this by describing multiple applications of lockable obfuscation. First, we show how to transform any attribute-based encryption (ABE) scheme into one in which the attributes used to encrypt the message are hidden from any user that is not authorized to decrypt the message. (Such a system is also know as predicate encryption with one-sided security.) The only previous construction due to Gorbunov, Vaikuntanathan and Wee is based off of a specific ABE scheme of Boneh et al. By enabling the transformation of any ABE scheme we can inherent different forms and features of the underlying scheme such as: multi-authority, adaptive security from polynomial hardness, regular language policies, etc.

We also show applications of lockable obfuscation to separation and uninstantiability results. We first show how to create new separation results in circular encryption that were previously based on indistinguishability obfuscation. This results in new separation results from learning with error including a public key bit encryption scheme that it IND-CPA secure and not circular secure. The tool of lockable obfuscation allows these constructions to be almost immediately realized by translation from previous indistinguishability obfuscation based constructions.

In a similar vein we provide random oracle uninstantiability results of the Fujisaki-Okamoto transformation (and related transformations) from the lockable obfuscation combined with fully homomorphic encryption. Again, we take advantage that previous work used indistinguishability obfuscation that obfuscated programs in a form that could easily be translated to lockable obfuscation.

## 1 Introduction

The topic of indistinguishability obfuscation has received an tremendous amount of attention from the cryptographic community over the last several years. Initially, the concept was introduced by Barak et al. [BGI+01, BGI+12] as an possible alternative to the notion of virtual black box obfuscation which they showed to be impossible to achieve for some functionalities. However, the concept indistinguishability obfuscation did not receive much immediate attention since (1) there were no such obfuscation candidates at the time and (2) the perceived lack of applications due to the fact that it only guaranteed security between two functionally equivalent circuits.

*Supported by NSF CNS-1228599 and CNS-1414082. DARPA through the U.S. Office of Naval Research under Contract N00014-11-1-0382, Google Faculty Research award, the Alfred P. Sloan Fellowship, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

In 2013, two works in the literature addressed these questions. First, Garg et al. [GGH+13b] provided the first indistinguishability obfuscation candidate using the Garg, Gentry and Halevi [GGH13a] multilinear map candidate. Then Sahai and Waters [SW14] introduced the "punctured programming" methodology for building cryptographic primitives from indistinguishability obfuscation which was used in their work and several subsequent works to resolve many open problems in cryptography.

When the potential of indistinguishability obfuscation was exposed, attention naturally moved to establishing security of obfuscation candidates since the original work of Garg et al. [GGH+13b] only provided a heuristic argument of security. Initial work in this line attempted to prove security under certain multilinear map models or assumptions [BGK+14, BR14, PST14, GLW14, GLSW15]. However, the security guarantees delivered from such proofs could only be as strong as the underlying multilinear map candidates [GGH13a, CLT13, GGH15, CLT15] which have been under a steady stream of cryptanalysis (see e.g. [CLT14, CHL+15, CGH+15, BGH+15, CLLT16, CLLT17, BWZ14, HJ16, Hal15, CFL+16, MSZ16, CJL16, ADGM16] and the references therein). To combat this there have been new multilinear map candidates proposed as well as models meant to capture most existing attacks [BMSZ16, GMM+16]. While these techniques present progress in defense against currently known cryptanalysis, it is unclear whether they can be connected to standard assumptions. Another set of works [BV15, AJ15, AJS15, Lin16a, LV16, Lin16b, AS16] have shown connections between certain types of functional encryption schemes and indistinguishability obfuscation with results showing that a constant degree multilnear maps combined with a constant depth PRG give indistinguishability obfuscation.

In this paper we approach the problem of achieving provably secure obfuscation from a different direction. Our philosophy is to anchor ourselves to the Learning with Errors (LWE) assumption and explore what applications and forms of obfuscation are achievable. Here we propose and define a new form of obfuscation that we call lockable obfuscation for which we give a construction that is provably secure under the LWE assumption. In addition, we show several applications of lockable obfuscation that were only known to this point under indistinguishability obfuscation.

We begin by informally introducing the notion of lockable obfuscation. A lockable obfuscation scheme consists of an obfuscation and evalution algorithms. The (randomized) obfuscation algorithm $\mathsf{Obf}$ takes as input a security parameter $\lambda$, a program $P$, a message $\mathsf{msg}$ and lock value $\alpha$ and outputs an obfuscated program $\widetilde{P}$. The evaluation algorithm $\mathsf{Eval}$ takes as input an obfuscated program $\widetilde{P}$ and an input $x$. If $P(x) = \alpha$ then the evaluation algorithm outputs the message $\mathsf{msg}$, where $P$, $\mathsf{msg}$ and $\alpha$ were the program, message and lock values input to create $\widetilde{P}$. Otherwise, if $P(x) \neq \alpha$ the evaluation algorithm (with high probability) outputs the $\perp$ symbol to indicate rejection.

Intuitively, security states that if the lock value is chosen at random and kept from the attacker, then the attacker cannot learn anything about the program and message other than their sizes. That is there exist a simulator $\mathsf{Sim}$ such that for all program message pairs $P, \mathsf{msg}$

$$\{\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha) : \alpha \leftarrow \{0,1\}^{m(\lambda)}\} \approx_c \mathsf{Sim}(1^\lambda, 1^{|P|}, 1^{|\mathsf{msg}|}).$$

We show how to construct a lockable obfuscation scheme for any polynomial sized circuit of sufficient output length from the Learning with Errors assumption. Our construction relies only on the polynomial hardness of the assumption (i.e. unlike witness encryption/indistinguishability obfuscation constructions, there is no sub-exponential hardness or complexity leveraging involved). [1] For this reason lockable obfuscation could be a preferred abstraction for building certain primitives even in a possible future where indistinguishability obfuscation is realizable from the assumption of LWE with subexponential hardness. The reason we don't require subexponential hardness of LWE is that the security of our construction is derived from the hidden lock value $\alpha$. In particular, our proof of security does *not* step through each possible input like many reductions to witness encryption or indistinguishability obfuscation [GLW14, GLSW15, BV15, AJ15, AJS15]. Also, note that since the lock value is chosen at random (and independent of the program), therefore lockable obfuscation could also be interpreted as obfuscation for a family of evasive functions [BBC+14].

[1] Note that we still require subexponential LWE modulus. This translates to a subexponential approximation factor in the worst case hardness of lattice problems.

We will defer the explanation of our construction and proof to the technical overview of Section 1.1 and move on to discussing applications.

**Hiding Attributes in Attribute-Based Encryption** In a (key-policy) Attribute-Based Encryption (ABE) system [SW05] a user will encrypt a message $\mathsf{msg}$ under an attribute string $x$. A private key, as issued by an authority, will be associated with a boolean function $f$. When a user holding a secret key for function $f$ attempts to decrypt a ciphertext he will learn the message if and only if $f(x) = 1$. Otherwise, the message remains hidden. While an ABE ciphertext will securely hide a private message, its security definition provides no guarantees about hiding the attribute string $x$. This can be problematic in many practical scenarios. Suppose we use ABE to encrypt email messages and use header metadata such as the sender, recipients, time and subject as attributes, where access functions are written over these attributes. In many settings this metadata itself will be very sensitive and disclosing it as part of the ciphertext is undesirable.

Almost all expressive[2] ABE systems built from standard tools[3] allow for an attacker to learn the attribute string associated with a ciphertext. One notable exception is the work of Gorbunov, Vaikuntanathan and Wee [GVW15] that achieves one-sided security in hiding attributes. In that their construction hides the attribute string $x$ as long as $f(x) = 0$ for all functions $f$ that the adversary has a secret key for. This gives a much improved security picture to before as in our example an attacker will not be able to learn the header metadata of the emails so long as they are not authorized to decrypt them. We will call such a scheme a predicate encryption scheme with one-sided security.

The GVW construction was built by adapting the ABE system of Boneh et al. [BGG$^+$14] and required an intricate knowledge of the original system in order to utilize certain special mathematical properties. The resulting construction achieves the same functionality of bounded depth circuits as the original as well as maintains selective security under the Learning with Errors assumption.

In this paper we show how to use lockable obfuscations (sometimes in combination with fully homomorphic encryption) to generically transform *any* ABE scheme into a predicate encryption scheme with one-sided security with corresponding functionality. An advantage of using a generic transformation is that it can take advantages of the features or forms of ABE constructions that have been introduced over the last 10+ years. including key-policy [GPSW06], ciphertext-policy [BSW07], adaptive security (without complexity leveraging) [LOS$^+$10], multi-authority [Cha07, CC09, LW11], and efficient expression of keys as deterministic finite automata [Wat12, Att14] and circuits [GVW13, BGG$^+$14]. Currently, there is no single ABE construction from standard tools that simultaneously delivers all such features. The most desirable ABE form can vary significantly between applications. Our transformations will allow one to inherent the properties of the underlying ABE scheme and at the same obtain one-sided hiding of the ciphertext attributes.

The technique for hiding attributes is fairly simple. To perform a (key-policy) encryption to attributes $x$ and message $\mathsf{msg}$ the encryption algorithm first chooses a random lock value $\alpha$. Next it creates a (sub) ciphertext $C$ which is an encryption of the message $\alpha$ under the attribute string $x$ in the original ABE system. Now consider the program $P$ which takes as input a secret key and then decrypts $C$ and outputs the result. The final ciphertext $\mathsf{ct}$ is a lockable obfuscation of this program $P$, under the lock value $\alpha$ and message $\mathsf{msg}$. Key generation is the same as in the original scheme and decryption is simply using the lockable obfuscation evaluation algorithm on the $\mathsf{ct}$ with the secret key as the input.

Correctness can be observed. Suppose a user has a secret key $\mathsf{sk}_f$ for function $f$ and applies it to a ciphertext encryption with attributes $x$ where $f(x) = 1$). The evaluation algorithm will output $\mathsf{msg}$ since $P(\mathsf{sk}_f) = \alpha$, which is the lock value. On the other hand suppose that the attacker does not have any secret keys for a function $f$ where $f(x) = 1$, then by the message hiding security of the underling ABE scheme, he won't be able to distinguish an encryption of the lock value $\alpha$ from an encryption of the all 0's string. Now that the lock value is hidden, one can take an additional hybrid step to argue that this is indistinguishable from a simulated obfuscation to erase knowledge of the program $P$ including $C$ and the attribute string $x$.

[2]We note that there are constructions of more limited expressivity such as vector matching [BW07] or inner product testing [KSW08] that achieve such security.

[3]For the purposes of this discussion we roughly consider number theoretic constructions grounded on RSA, bilinear maps and (Ring) LWE to be standard tools and those based on multilinear maps or indistinguishability obfuscation not to be.

The above construction works for any scheme with an apriori bounded size decryption circuit. By adding a level of indirection one can leverage leveled homomorphic encryption (which is realizable under LWE [BV11, BGV12]) to upgrade this to any scheme with an apriori decryption circuit depth. Or leverage fully homomorphic encryption to work without any depth bounds. The details of the transformation are given in Section 5.

**Separation and Uninstantiability Results** We now demonstrate the power of lockable obfuscation for achieving negative results in cryptography by focusing on two families of separation and uninstantiability results.

In recent years there has been a significant interest on the problem of circular security [CL01, Lau02, ABHS09] perhaps in large part due to Gentry's [Gen09] result showing that a leveled homomorphic encryption scheme that is circular secure implies unbounded fully homomorphic encryption. Roughly, an encryption system is circular secure if it maintains security in the setting where there are $n$ pairs of public keys and ciphertexts arranged such that the $i^{th}$ ciphertext encrypts the $(i+1)^{th}$ secret key.

There have been several separation results [ABBC10, CGH12, Rot13, KRW15, MO14, BHW15, AP16, KW16, GKW17a, GKW17b] showing that such security does not come for free. In particular, they show in different contexts that there exist schemes that are IND-CPA secure, but not circular secure. (We discuss this prior work in detail in Section 6). A natural dichotomy of these results is between separations achieved using indistinguishability obfuscation [KRW15, GKW17a] and those built from standard assumptions [ABBC10, CGH12, BHW15, AP16, KW16, GKW17b] such as assumptions on bilinear groups or LWE.

Separations built on indistinguishability obfuscation have the advantage that they can be developed relatively rapidly and are also relatively simple as one can build constructions using "normal" programs without diving into number theory. The disadvantage is that indistinguishability is not currently known from any standard assumptions. On the flip side, the number theoretic constructions are based on much more standard assumptions. However, they developing and understanding such solutions is a much more arduous task.

In Section 6 we show how to translate existing two indistinguishability obfuscation separation results due to Koppula, Ramchen and Waters [KRW15] and one result due Goyal, Koppula and Waters [GKW17a] to rely on lockable obfuscation. The translations are extremely straightforward and our resulting solutions are almost identical with the exception that we use lockable obfuscation. The main insight is that the programs obfuscated in the above results come in a lockable friendly form. In particular, they perform a sequence of computations on the input that result in a value $s$. Then the program tests and reports if $\mathrm{PRG}(s) = t$ for a pseudorandom generator PRG and hardwired value $t$. Using lockable obfuscation one simply uses $s$ as the output (which is a possible lock value).

Our translations of these prior works to lockable obfuscation lead to separations that can be built on the Learning with Errors assumption. Concretely, we obtain new results from LWE that were previously known only from indistinguishability obfuscation. (1) We show how to build a *public key* bit encryption scheme that is not circular secure and (2) we show a separation for unbounded length cycles.

Our second family of negative results relate to a grouping of constructions related to the well known Fujisaki-Okamoto transformation that achieves chosen ciphertext security in the random oracle model from any scheme which is IND-CPA secure. Included in this grouping are: the Bellare et al. [BHSV98] transformation from an IND-CPA scheme to an injective trapdoor function, two transformations from IND-CPA to IND-CCA security due to Fujisaki and Okamoto [FO99a, FO99b] and the deterministic encryption construction of Bellare, Boldyreva and O'Neill [BBO07].

All of the constructions follow a similar paradigm where they encrypt a string $x$ under random coins determined from $H(x)$. (How the string $x$ is construed varies somewhat between the schemes.) The works above show that if $H$ is presented as an oracle access to a random function, the transformation results in a secure scheme under the relevant definition.

We give a random oracle uninstantiability [CGH98] result where using lockable obfuscation there exists an encryption scheme where for any hash function of up to apriori bounded size the applying the above transformations will result in an insecure encryption scheme — the message will be easily discovereable. If

we add the assumption of fully homomorphic encryption we can remove the bounded size restriction.

Brzuska, Farshim and Mittelbach [BFM15] achieved these results using indistinguishability obfuscation.[4] We realize our results by simply translating the BFM result to move from indistinguishability obfuscation to lockable obfuscation. Again, this is possible because the programs obfuscated in the BFM paper follow the same lockable friendly form.

**Indistinguishability Obfuscation for Rejecting Programs** For our final application we now consider a new notion of obfucation that we call indistinguishability obfuscation for rejecting programs and show how to construct it from lockable obfuscation and witness encryption [GGSW13] for circuit satisfiability.

Obfuscators that meet this notion will be defined over boolean circuits. Like indistinguishability obfuscation our obfuscator will take in any (not necessarily rejecting) boolean circuit $C$ in a class and output an obfuscated program that is functionally equivalent to $C$. However, the security guarantees given by such an obfuscator are limited to "rejecting" programs. Informally, they state that no PPT adversary can distinguish between circuits $C_0$ and $C_1$ so long as for all inputs $x$ $C_0(x) = C_1(x) = 0$. In contrast, standard indistinguishability obfuscation security allows $C_0, C_1$ to have arbitrary (both 0 and 1) outputs so long as they are functionally equivalent.

Our construction is simple and follows along the same conceptual lines as our techniques for building predicate encryption with one sided security from Attribute-Based Encryption.

**Concurrent Work** In an independent and concurrent work, Wichs-Zirdelis proposed a similar notion called *Obfuscation for Compute-and-Compare Programs*. While the notions are very similar, the syntax is different. A compute-and-compare program $CC[f, y]$ is defined by a function $f$ and a value $y$. Obfuscation of $CC[f, y]$ outputs a program $P$ such that $P$, on any input $x$, outputs 1 if $f(x) = y$. For security, [WZ17] require that for a randomly chosen $y$ (from a high-entropy distribution), the obfuscation of $CC[f, y]$ is indistinguishable from simulated obfuscation, where the simulator gets only size of $f$ and $y$. Wichs-Zirdelis also extend this notion where the obfuscation algorithm also takes as input a message $m$ together with $f, y$, and the obfuscated program $P$ on input $x$, outputs $m$ if $f(x) = y$. Security requires that the message $m$ is also hidden. This message-based version of obfuscation of compute-and-compare programs is identical to our notion of lockable obfuscation, modulo the distribution of lock $y$, which is uniform in our case and can be any high-entropy distribution in their case. However, this gap can be simply bridged by using an PRG for pseudo-entropy seeds instead of a regular PRG.

Both works have a few applications in common. This includes the ABE to predicate encryption transformation, witness encryption to reject-iO and circular security counterexamples. In addition, we show new uninstantiability results in the random oracle model. Applications unique to [WZ17] are transformation from any secure sketch [DORS08] to private secure sketch [DS05], and obfuscation for conjunctions/affine testers.

## 1.1 Overview of our Lockable Obfuscation Construction

We will now describe our lockable obfuscation scheme for a family of poly-depth circuits. The construction is described in details in Section 4. At a high level, our scheme can be divided into three components — (1) A lockable obfuscation scheme for a family of low-depth circuits and 1-bit messages, (2) a bootstrapping mechanism to amplify to lockable obfuscation for a family of poly-depth circuits and 1-bit messages, and (3) extending to lockable obfuscation for a family of poly-depth circuits and multi-bit messages. (Note in our actual construction, we combine the first two components into one for technical reasons.)

**Lockable Obfuscation for Low-Depth Circuits and 1-Bit Messages.** The primary ingredients of our construction are low-depth pseudorandom generators (PRGs), lattice trapdoors [GPV08], telescoping products/cascading cancellations [KW16, AP16, GGH15] and oblivious sequence transformation [GKW17b].

[4] Technically, their result with no bounds on the hash function required indistinguishability obfuscation for Turing Machines with unbounded input. However, this could have been replaced with indistinguishability obfuscation for circuits and fully homomorphic encryption.

First, let us recall the notion of permutation branching programs, lattice trapdoors and oblivious sequence transformation. A permutation branching program of length $L$ and width $w$ can be represented using $w$ states, $2L$ permutations $\sigma_{j,b}$ over states for each level $j \leq L$, an input-selector function $\mathsf{inp}(\cdot)$ which determines the input read at each level, and an accepting and rejecting state. The program execution starts at state 1 of level 0. Suppose the branching program reads first input bit (say $b$) at level 1 (i.e., $\mathsf{inp}(1) = 1$). Then, the state of the program changes to $\sigma_{1,b}(\mathsf{st})$. Such a process can be carried out (iteratively) to compute the final program state at level $L$. Depending upon the final state, the program either accepts or rejects.

A lattice trapdoor generation algorithm can be used to sample a (uniformly looking) matrix $\mathbf{A}$ together with a trapdoor $T_{\mathbf{A}}$. The trapdoor can be used to compute, for any matrix $\mathbf{U}$, a low norm matrix $\mathbf{S}$ such that $\mathbf{A} \cdot \mathbf{S} = \mathbf{U}$.[5] As a result, the matrix $\mathbf{S}$ can be used to 'transform' any matrix $\widetilde{\mathbf{A}} \approx \mathbf{A}$ to another matrix $\widetilde{\mathbf{U}} \approx \mathbf{U}$. Oblivious sequence transformation is a technique that enables sampling a sequence of matrices $\mathbf{B}_1, \ldots, \mathbf{B}_w$ along with some trapdoor such that for any sequence of matrices $\mathbf{U}_1, \ldots, \mathbf{U}_w$, one can construct a short matrix $\mathbf{X}$ such that $\mathbf{B}_i \cdot \mathbf{X} = \mathbf{U}_i$ for all $i$ (and $\mathbf{X}$ is oblivious of $i$).

Moving on to our construction, at a high level the obfuscator starts by first generating a sequence of permutation branching programs corresponding to the circuit $C$ (where each branching program computes one output bit), and then encoding the state transition permutations for each level for every branching program using the technique of oblivious sequence transformation. Let $\ell$ be the output length of $C$. In other words, for obfuscating circuit $C$ under lock $\alpha$ with message $\mathsf{msg}$, the obfuscator first expresses $C$ as a set of width 5 permutation branching programs $\{\mathsf{BP}^{(i)}\}_i$ of polynomial length $L$, where for each $i \in [\ell]$ $\mathsf{BP}^{(i)}$ computes the $i^{th}$ output bit of circuit $C$.[6] Without loss of generality, we can assume that all branching programs have a common input selector function $\mathsf{inp}(\cdot)$ such that $\mathsf{inp}(j)$ bit of the input is read at level $j$. The obfuscation algorithm continues by choosing $5(L+1)\ell$ matrices, one matrix for each (level, state) of each branching program.[7] Let $\mathbf{B}^{(i)}_{j,w}$ be the matrix corresponding to state $w \in [5]$ at level $j \leq L$ for branching program $\mathsf{BP}^{(i)}$, $i \in [\ell]$. For every $i \in [\ell]$, the matrices $\{\mathbf{B}^{(i)}_{j,1}, \ldots, \mathbf{B}^{(i)}_{j,5}\}$ for the first $L$ levels (i.e., all but top-level matrices) are sampled such that they have a common trapdoor $T^{(i)}_j$, i.e. using oblivious sequence transformation. The top-level matrices, however, are sampled uniformly at random without a trapdoor subject to the constraint that the top-level matrices corresponding to lock string $\alpha$ sum to a special fixed matrix depending upon the message $\mathsf{msg}$. More formally, for each $i \in [\ell]$, let $q$ be the LWE modulus, and $\mathsf{acc}^{(i)}$ and $\mathsf{rej}^{(i)}$ be the accepting and rejecting states for $\mathsf{BP}^{(i)}$, then the obfuscator chooses the matrices $\mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}}, \mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}}$ such that

$$\sum_{i\,:\,\alpha_i=0} \mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}} + \sum_{i\,:\,\alpha_i=1} \mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}} = \begin{cases} \mathbf{0}^{n\times m} & \text{if } \mathsf{msg}=0. \\ \sqrt{q}\cdot\left[\mathbf{I}_n \,||\, \mathbf{0}^{n\times(m-n)}\right] & \text{if } \mathsf{msg}=1. \end{cases}$$

For each level $j \in [L]$, the obfuscation algorithm also chooses two low-norm matrices $\mathbf{S}^{(0)}_j$ and $\mathbf{S}^{(1)}_j$ (these are shared across all branching programs), and computes $2\ell$ low-norm matrices $\{\mathbf{C}^{(i,0)}_j, \mathbf{C}^{(i,1)}_j\}_{i,j}$ such that for every state $w \in [5]$, $\mathbf{B}^{(i)}_{j-1,w} \cdot \mathbf{C}^{(i,0)}_j \approx \mathbf{S}^{(0)}_j \cdot \mathbf{B}^{(i)}_{j,\sigma^{(i)}_{j,0}(w)}$ and $\mathbf{B}^{(i)}_{j-1,w} \cdot \mathbf{C}^{(i,1)}_j \approx \mathbf{S}^{(1)}_j \cdot \mathbf{B}^{(i)}_{j,\sigma^{(i)}_{j,1}(w)}$. That is, the matrices $\mathbf{C}^{(i,0)}_j$ and $\mathbf{C}^{(i,1)}_j$ represent the state transition from level $j-1$ to $j$ when bit 0 or 1 is read at step $j$ of branching program execution. For each $i \in [\ell], j \in [L]$, the $\mathbf{C}^{(i,b)}_j$ matrices can be generated using the lattice trapdoors $T^{(i)}_j$. The obfuscation algorithm outputs these matrices $\{\mathbf{C}^{(i,0)}_j, \mathbf{C}^{(i,1)}_j\}_{i,j}$ together with the base-level matrices $\{\mathbf{B}^{(i)}_{0,1}\}_i$ as the final obfuscated program.

At a high level, one could visualize the obfuscated program which consists of the base-level matrices

[5] For ease of exposition, we will use the notation $\mathbf{A}^{-1}(\cdot)$ to represent the pre-image operation. In the formal description of our algorithms later, we use the pre-image sampling algorithm $\mathsf{SamplePre}$.

[6] From Barrington's Theorem [Bar86], we know that for every $\mathbf{NC}^1$ circuits there exists a width 5 permutation branching program of polynomial length.

[7] Note that if a branching program has length $L$, then it has $L+1$ levels.

$\{\mathbf{B}_{0,1}^{(i)}\}_i$ and matrices $\{\mathbf{C}_j^{(i,0)}, \mathbf{C}_j^{(i,1)}\}_{i,j}$ as "encodings" of the branching program starting states and state transition permutations, respectively. Therefore, evaluating an obfuscated program on some input $x$ will be analogous to evaluating the branching programs $\mathsf{BP}^{(i)}$ on input $x$ directly. Fix some $i \in [\ell]$. Suppose the first input bit $x_1$ is read at level 1. Then evaluation of $\mathsf{BP}^{(i)}$ at level 1 would map the state 1 at level 0 to state $\sigma_{1,x_1}^{(i)}$ at level 1. Analogously, the obfuscation evaluator can compute $\mathbf{B}_{0,1}^{(i)} \cdot \mathbf{C}_1^{(i,x_1)} \approx \mathbf{S}_1^{(x_1)} \cdot \mathbf{B}_{1,\sigma_{1,x_1}^{(i)}}^{(i)}$. In general, if the program state at level $j-1$ during execution is $w$, then the evaluator will accumulate the product of the form $\mathbf{\Gamma}_{j-1} \cdot \mathbf{B}_{j,w}^{(i)}$, where $\mathbf{\Gamma}_{j-1}$ is a product of $j-1$ low-norm matrices. This can be easily verified as follows. Suppose the next bit read is $b$, then the new state at level $j$ will be $\sigma_{j,b}^{(i)}(w)$, thus the new accumulated product during obfuscation evaluation will be $\mathbf{\Gamma}_{j-1} \cdot \mathbf{B}_{j,w}^{(i)} \cdot \mathbf{C}_j^{(i,b)} \approx \mathbf{\Gamma}_j \cdot \mathbf{B}_{j+1,\sigma_{j,b}^{(i)}(w)}^{(i)}$, where $\mathbf{\Gamma}_j = \mathbf{\Gamma}_{j-1} \cdot \mathbf{S}_j^{(b)}$. Therefore, the invariant is maintained. Note that the matrix $\mathbf{\Gamma}_L$ will be same for all branching programs since the low-norm matrices $\mathbf{S}_j^{(0)}$ and $\mathbf{S}_j^{(1)}$ are shared across all branching programs.

Continuing this way, the evaluator can iteratively compute the matrix product at the top. Thus, for each branching program, the accumulated product at the top will either be $\approx \mathbf{\Gamma}_L \cdot \mathbf{B}_{L,\mathsf{acc}^{(i)}}^{(i)}$ or $\approx \mathbf{\Gamma}_L \cdot \mathbf{B}_{L,\mathsf{rej}^{(i)}}^{(i)}$, depending on whether $C(x)_i = 0$ or 1. Let $\mathbf{\Delta}^{(i)} = \mathbf{\Gamma}_L \cdot \mathbf{B}_{L,\mathsf{st}^{(i)}}^{(i)}$, where $\mathsf{st}^{(i)} = \mathsf{acc}^{(i)}$ or $\mathsf{rej}^{(i)}$ depending on $C(x)_i$. Finally, the evaluator simply sums the top-level accumulated products ($\approx \mathbf{\Delta}^{(i)}$) and checks whether the norm of the final summed matrix lies in appropriate range. More concretely, consider the case when $C(x) = \alpha$ and $\mathsf{msg} = 0$, then $\sum_i \mathbf{\Delta}^{(i)} = \Gamma_L \cdot \sum_i \mathbf{B}_{L,\mathsf{st}^{(i)}}^{(i)} = \mathbf{0}^{n \times m}$. Since the final top-level matrix sum is close to $\sum_i \mathbf{\Delta}^{(i)}$, thus it will have norm close to 0, and hence the evaluator can simply test this and output 0 as the message.

Similarly we could argue correctness for the cases when $\mathsf{msg} = 1$ or $C(x) \neq \alpha$. However, our current proof techniques do not seem sufficient for proving the security of above construction. The reason is that there is an inherent tension in setting scheme parameters while basing security on LWE. This is discussed in detail later in Section 4.3. We were able to bypass this problem by obfuscating an "expanded" circuit, which evaluates a low-depth pairwise independent hash function $h$ and a low-depth pseudorandom generator PRG with a large enough polynomial stretch (in succession) on the output of circuit $C$, instead of directly obfuscating circuit $C$ using the above matrix encoding procedure.

In other words, let $Q$ be the circuit that on input $x$, outputs $\mathrm{PRG}(h(C(x)))$. Observe that if $h$ and PRG can be computed by low-depth circuits ($\mathbf{NC}^1$), then $Q$ also can be computed by an $\mathbf{NC}^1$ circuit (since $C$ is assumed to be a log-depth circuit). Therefore, $Q$ can be expressed by a set of width 5 permutation branching programs of polynomial length $L$ as well. Additionally, now the matrix component generation procedure will use $\beta = \mathrm{PRG}(h(\alpha))$ as the lock instead of $\alpha$. This modification is sufficient to avoid the tension between scheme parameters, and also allows us to prove security of our scheme under LWE with only polynomial hardness. This completes the description of our lockable obfuscation scheme for low-depth circuits and 1-bit messages.

**Bootstrapping Lockable Obfuscation.** Let $\mathcal{O}_{\mathsf{NC}^1}$ be a lockable obfuscator for log-depth circuits. We will use leveled homomorphic encryption (LHE) with an $\mathbf{NC}^1$ decryption circuit to bootstrap $\mathcal{O}_{\mathsf{NC}^1}$ to an obfuscator that works for any depth $d$. The obfuscator gets as input a circuit $C$ of depth $d$, a string $\alpha$ and a message $\mathsf{msg}$. It first chooses the LHE secret-evaluation keys and encrypts the circuit $C$. Let $Q$ be the circuit which takes as input an LHE ciphertext and decrypts it using the hardwired LHE secret key and outputs the decrypted string. Note that the circuit $Q$ is a logarithmic depth circuit. The obfuscator outputs $\mathcal{O} \leftarrow \mathcal{O}_{\mathsf{NC}^1}(Q, \alpha, \mathsf{msg})$ together with the encryption of $C$ and the LHE evaluation key.

*Evaluating on input $x$.* Let $U_x(\cdot)$ be the universal circuit with input $x$ hardwired (that is, it takes a circuit $C$ as input and outputs $C(x)$). The evaluation algorithm first homomorphically evaluates the circuit $U_x$ on the encryption of $C$. This results in an LHE ciphertext $\mathsf{ct}$, which is an encryption of $C(x)$. It then evaluates the obfuscation $\mathcal{O}$ on input $\mathsf{ct}$, and outputs the resulting string. Using the correctness of $\mathcal{O}$ and the LHE scheme, we can argue that if $C(x) = \alpha$, then the evaluation outputs $\mathsf{msg}$, and it outputs $\perp$ otherwise.

The security proof here is fairly simple. Using the security of the underlying obfuscator $\mathcal{O}_{\mathsf{NC}^1}$, we first switch the obfuscation $\mathcal{O}$ to be a simulated obfuscation. Once the obfuscator $\mathcal{O}$ is simulated, the obfuscator no longer needs the LHE secret key. Therefore, we can now replace the LHE encryption of $C$ with encryption of zeros, thereby erasing all the information about circuit $C$ except its size. Therefore, the final simulator simply outputs an encryption of zeros, together with a simulated obfuscation, and this is indistinguishable from the honestly computed obfuscation.

**Extending Lockable Obfuscation for 1-Bit Messages to Multi-Bit Messages.** We would like point out that the standard repitition method for extending message space does not work for lockable obfuscation schemes because the security is only guaranteed when the adversary does not know the lock string $\alpha$. However, we observe there are still multiple ways to extend its message space. We briefly discuss two possible extensions. One option could be to encode a multi-bit message directly in the top-level matrices. Currently, the top-level matrices are set to sum to $\mathbf{0}^{n\times m}$ if $\mathsf{msg} = 0$, otherwise to $\sqrt{q}\cdot\left[\mathbf{I}_n \,||\, \mathbf{0}^{n\times(m-n)}\right]$. However, if we interpret the mesage $\mathsf{msg}$ as an integer $v < \sqrt{q}/2$, then we could simply set the sum to be $v\cdot\sqrt{q}\cdot\left[\mathbf{I}_n \,||\, \mathbf{0}^{n\times(m-n)}\right]$.

The second extension could be carried more generally without exploiting the mathematical structure of the underlying obfuscation scheme. The high level idea is to again "expand" the circuit $C$ using a pairwise independent hash function and a pseudorandom generator before obfuscation. Suppose the lock $\alpha$ be a string of length $k$. Let $\beta = \mathrm{PRG}(h(\alpha))$ and $|\beta| = \ell\cdot k$. To obfuscate circuit $C$ under lock $\alpha$ for an $\ell$-bit message $\mathsf{msg}$, the multi-bit obfuscator (for each $i < \ell$) independently obfuscates the circuit $Q[i]$ under lock $\beta[i]$ for message $\mathsf{msg}_i$ using the 1-bit obfuscation scheme, where circuit $Q[i]$ denotes the circuit that outputs the $i\cdot k+1,\ldots,(i+1)\cdot k$ output bits of circuit $\mathrm{PRG}(h(C(\cdot)))$ and $\beta[i] = \beta_{i\cdot k+1},\ldots,\beta_{(i+1)\cdot k}$. The security proof follows from a simple hybrid argument. This transformation is described later in Appendix C.

This completes the technical overview of our lockable obfuscation scheme.

## 1.2 More on Related Encoding Works

The idea of using lattice trapdoors for constructing multilinear maps was first seen in the work of Gentry, Gorbunov and Halevi (GGH) where they proposed a candidate for graph-induced multilinear maps [GGH15]. Their work builds upon the homomorphic encryption scheme of Gentry, Sahai and Waters [GSW13] which could be considered as the starting point of cascading cancellations technique. In [GGH15], there is a fixed (directed acyclic) graph $G = (V, E)$, and plaintext messages are associated with edges. Each vertex $u$ has an associated matrix $\mathbf{A}_u$ and a secret parameter $T_u$ which is the trapdoor for $\mathbf{A}_u$. The plaintext space consists of matrices, and the encoding of a matrix $\mathbf{M}$ along an edge $(u, v)$ is $\mathbf{A}_u^{-1}(\mathbf{M}\cdot\mathbf{A}_v + \mathsf{noise})$. Note that given an encoding of matrix $\mathbf{M}_1$ for edge $(u, v)$ and an encoding of matrix $\mathbf{M}_2$ for edge $(v, w)$, one can compute an encoding of $\mathbf{M}_1\cdot\mathbf{M}_2$ along path $(u, v, w)$.Informally, their intuition was that given all the encodings it should be easy to compute an encoding for any path in graph $G$, however it should remain hard for anyone to generate encodings over any non-path combination of vertices. Using these multilinear maps, GGH gave candidate constructions for obfuscation and multipartite key-agreement. However, there were no security proofs for these candidates, and their key-agreement protocol was later shown to be broken [CLLT16]. In a later work, Brakerski et al. [BVWW16] gave a construction for obfuscating conjunctions based on an entropic variant of Ring-LWE. In that work, they observed that if the underlying graph was a straight-line graph (i.e., an ordered sequence of vertices), then the corresponding GGH multilinear maps provide some provable security properties.

In a more recent work, Koppula and Waters [KW16] used what they called cascading cancellations technique for constructing $k$-circular security separations. Their construction deviates from the GGH/BVWW paradigm in two crucial aspects. First, their construction involves multiple *strands* (say $\ell$) of length $k$, instead of just a single strand. Unlike the previous works, which involved directly comparing two distinct encodings (or, components) along the same path for equality, they first combined (using matrix multiplications) all the

$k$ components for each different strand, and then summed the $\ell$ final components (one for each strand) to perform an equality check with a fixed value. Second, what is mechanically close to the "plaintext" encoded values in GGH is viewed here as simply random matrices (and just part of the overall randomness) and not a value to be encoded. This differs from GGH in which the the elements being encoded were labeled as "plaintext" values. For instance, the elements that were encoded in the GGH obfuscation candidate were the state transition matrices of the actual branching program beign obfuscated and thus can reflect some stronger semantics. Concurrently, Alamati and Peikert [AP16] also provided circular security separations, that had a cancellation type effect although with a different technical approach.

Most recently, Goyal, Koppula and Waters [GKW17b] further advanced the existing techniques for constructing bit-circular security separations. One of their most important contributions was an alternative mechanism to encode and *hide* (branching) programs using lattice trapdoors. To this end, they introduced a novel technique which they call oblivious sequence transformation. This is a significant departure from prior works as previously most works generated the matrix components (or encodings) independently, however it was essential in their construction that the components be jointly generated. At a high level, they provided new techniques to encode and hide a permutation between a sequence of nodes. Informally, they gave a mechanism to encode a permutation between nodes $u_1, \ldots, u_5$ and $w_1, \ldots, w_5$ such that given a node $u_i$ one could obliviously go to its corresponding node $w_j$. Another important aspect of their work was to encode a log-depth Pseudo Random Generator (PRG) using the oblivious sequence transformation technique such that the PRG could be publicly evaluated and if the output of computation is some fixed (but unknown) value, then it could be efficiently tested. This seems to be one of the most important technical aspect of their work since in order to prove security from the LWE assumption as well as guarantee efficient testing, encoding a log-depth PRG with a large polynomial stretch is essential. We finally remark that Canetti and Chen [CC17] recently used the LWE assumption to achieve 1-collusion secure constrained PRFs for **NC1** with constraint hiding. Their construction involves a single strand like BVWW and their embedding of a branching program follows the GGH style of putting branching programs into the encoded plaintext values much more closely.

# 2 Preliminaries

**Notations.** Let PPT denote probabilistic polynomial-time. We will use lowercase bold letters for vectors (e.g. $\mathbf{v}$) and uppercase bold letters for matrices (e.g. $\mathbf{A}$). For any integer $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers modulo $q$. We represent $\mathbb{Z}_q$ as integers in the range $(-q/2, q/2]$. For a vector $\mathbf{v}$, we let $\|\mathbf{v}\|$ denote its $\ell_2$ norm and $\|\mathbf{v}\|_\infty$ denote its infinity norm. Similarly, for matrices $\|\cdot\|$ and $\|\cdot\|_\infty$ denote their $\ell_2$ and infinity norms (respectively).

We denote the set of all positive integers upto $n$ as $[n] := \{1, \ldots, n\}$. For any finite set $S$, $x \leftarrow S$ denotes a uniformly random element $x$ from the set $S$. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element $x$ drawn from distribution $\mathcal{D}$. The distribution $\mathcal{D}^n$ is used to represent a distribution over vectors of $n$ components, where each component is drawn independently from the distribution $\mathcal{D}$. Two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, parameterized by security parameter $\lambda$, are said to be computationally indistinguishable, represented by $\mathcal{D}_1 \approx_c \mathcal{D}_2$, if for all PPT adversaries $\mathcal{A}$, $|\Pr[1 \leftarrow \mathcal{A}(x) : x \leftarrow \mathcal{D}_1] - \Pr[1 \leftarrow \mathcal{A}(x) : x \leftarrow \mathcal{D}_2]| \leq \text{negl}(\lambda)$. For a family of distributions $\chi = \{\chi(\lambda)\}_\lambda$ over the integers, and integers bounds $B = \{B(\lambda)\}_\lambda$, we say that $\chi$ is $B$-bounded if $\Pr[|x| \leq B(\lambda) : x \leftarrow \chi(\lambda)] = 1$.
This section closely follows [GKW17b].

**Min-Entropy and Randomness Extraction.** The min-entropy of a random variable $X$ is defined as $\mathbf{H}_\infty(X) \stackrel{\text{def}}{=} -\log_2(\max_x \Pr[X = x])$. Let $\mathsf{SD}(X, Y)$ denote the statistical distance between two random variables $X$ and $Y$. Below we state the Leftover Hash Lemma (LHL) from [HILL99, DRS04, DORS08].

**Theorem 2.1.** Let $\mathcal{H} = \{h : X \to Y\}_{h \in \mathcal{H}}$ be a universal hash family, then for any random variable $W$

taking values in $X$, the following holds

$$\mathsf{SD}\left((h, h(W)), (h, U_Y)\right) \leq \frac{1}{2}\sqrt{2^{-\mathbf{H}_\infty(W)} \cdot |Y|}\ .$$

We will use the following corollary, which follows from the Leftover Hash Lemma.

**Corollary 2.1.** Let $\ell > m \cdot n \log_2 q + \omega(\log n)$ and $q$ a prime. Let $\mathbf{R}$ be an $k \times m$ matrix chosen as per distribution $\mathcal{R}$, where $k = k(n)$ is polynomial in $n$ and $\mathbf{H}_\infty(\mathcal{R}) = \ell$. Let $\mathbf{A}$ and $\mathbf{B}$ be matrices chosen uniformly in $\mathbb{Z}_q^{n \times k}$ and $\mathbb{Z}_q^{n \times m}$, respectively. Then the statistical distance between the following distributions is negligible in $n$.

$$\{(\mathbf{A}, \mathbf{A} \cdot \mathbf{R})\} \approx_s \{(\mathbf{A}, \mathbf{B})\}$$

*Proof.* The proof of above corollary follows directly from the Leftover Hash Lemma. Note that for a prime $q$ the family of hash functions $h_{\mathbf{A}} : \mathbb{Z}_q^{k \times m} \to \mathbb{Z}_q^{n \times m}$ for $\mathbf{A} \in \mathbb{Z}_q^{n \times k}$ defined by $h_{\mathbf{A}}(\mathbf{X}) = \mathbf{A} \cdot \mathbf{X}$ is universal. Therefore, if $\mathcal{R}$ has sufficient min-entropy, i.e. $\ell > m \cdot n \log_2 q + \omega(\log n)$, then the Leftover Hash Lemma states that statistical distance between the distributions $(\mathbf{A}, \mathbf{A} \cdot \mathbf{R})$ and $(\mathbf{A}, \mathbf{B})$ is at most $2^{-\omega(\log n)}$ which is negligible in $n$ as desired. ∎

## 2.1 Lattice Preliminaries

An $m$-dimensional lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^m$. Given positive integers $n, m, q$ and a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we let $\Lambda_q^\perp(\mathbf{A})$ denote the lattice $\{\mathbf{x} \in \mathbb{Z}^m \ : \ \mathbf{A} \cdot \mathbf{x} = \mathbf{0} \mod q\}$. For $\mathbf{u} \in \mathbb{Z}_q^n$, we let $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ denote the coset $\{\mathbf{x} \in \mathbb{Z}^m \ : \ \mathbf{A} \cdot \mathbf{x} = \mathbf{u} \mod q\}$.

**Discrete Gaussians.** Let $\sigma$ be any positive real number. The Gaussian distribution $\mathcal{D}_\sigma$ with parameter $\sigma$ is defined by the probability distribution function $\rho_\sigma(\mathbf{x}) = \exp(-\pi \|\mathbf{x}\|^2 / \sigma^2)$. For any set $\mathcal{L} \subset \mathcal{R}^m$, define $\rho_\sigma(\mathcal{L}) = \sum_{\mathbf{x} \in \mathcal{L}} \rho_\sigma(\mathbf{x})$. The discrete Gaussian distribution $\mathcal{D}_{\mathcal{L},\sigma}$ over $\mathcal{L}$ with parameter $\sigma$ is defined by the probability distribution function $\rho_{\mathcal{L},\sigma}(\mathbf{x}) = \rho_\sigma(\mathbf{x})/\rho_\sigma(\mathcal{L})$ for all $\mathbf{x} \in \mathcal{L}$.

The following lemma (Lemma 4.4 of [MR07], [GPV08]) shows that if the parameter $\sigma$ of a discrete Gaussian distribution is small, then any vector drawn from this distribution will be short (with high probability).

**Lemma 2.1.** Let $m, n, q$ be positive integers with $m > n$, $q \geq 2$. Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be a matrix of dimensions $n \times m$, $\sigma = \tilde{\Omega}(n)$ and $\mathcal{L} = \Lambda_q^\perp(\mathbf{A})$. Then

$$\Pr[\|\mathbf{x}\| > \sqrt{m} \cdot \sigma : \mathbf{x} \leftarrow \mathcal{D}_{\mathcal{L},\sigma}] \leq \mathrm{negl}(n).$$

**Truncated Discrete Gaussians.** The truncated discrete Gaussian distribution over $\mathbb{Z}^m$ with parameter $\sigma$, denoted by $\widetilde{\mathcal{D}}_{\mathbb{Z}^m,\sigma}$, is same as the discrete Gaussian distribution $\mathcal{D}_\sigma$ except it outputs 0 whenever the $\ell_\infty$ norm exceeds $\sqrt{m} \cdot \sigma$. Note that, by definition, $\widetilde{\mathcal{D}}_{\mathbb{Z}^m,\sigma}$ is $\sqrt{m} \cdot \sigma$-bounded. Also, note that $\widetilde{\mathcal{D}}_{\mathbb{Z}^m,\sigma} \approx_s \mathcal{D}_{\mathbb{Z}^m,\sigma}$.

**Learning with Errors (LWE).** The Learning with Errors (LWE) problem was introduced by Regev [Reg05]. The LWE problem has four parameters: the dimension of the lattice $n$, the number of samples $m$, the modulus $q$ and the error distribution $\chi = \chi(n)$.

**Assumption 1** (Learning with Errors)**.** Let $n$, $m$ and $q$ be positive integers and $\chi$ a noise distribution over $\mathbb{Z}_q$. The Learning with Errors assumption $(n, m, q, \chi)$-$\mathsf{LWE}$, parameterized by $n, m, q, \chi$, states that the following distributions are computationally indistinguishable:

$$\left\{(\mathbf{A}, \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}^\top) \ : \ \begin{array}{l} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \\ \mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi^m \end{array}\right\} \approx_c \left\{(\mathbf{A}, \mathbf{u}^\top) \ : \ \begin{array}{l} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \\ \mathbf{u} \leftarrow \mathbb{Z}_q^m \end{array}\right\}$$

Under a quantum reduction, Regev [Reg05] showed that for certain noise distributions, LWE is as hard as worst case lattice problems such as the decisional approximate shortest vector problem ($\mathsf{GapSVP}$) and approximate shortest independent vectors problem ($\mathsf{SIVP}$). Later works [Pei09, BLP+13] showed classical reductions from LWE to $\mathsf{GapSVP}_\gamma$.

These works show that for $B$-bounded discretized Gaussian error distributions $\chi$, solving $(n, m, q, \chi)$-$\mathsf{LWE}$ is as hard as approximating $\mathsf{GapSVP}_\gamma$ and $\mathsf{SIVP}_\gamma$ to a factor of $\tilde{O}(n \cdot q/B)$. Given the current state of art in lattice algorithms, $\mathsf{GapSVP}_\gamma$ and $\mathsf{SIVP}_\gamma$ are believed to be hard for $\gamma = \tilde{O}(2^{n^\epsilon})$ (for fixed $\epsilon \in (0, 1/2)$), and therefore $(n, m, q, \chi)$-$\mathsf{LWE}$ is believed to be hard for $B$-bounded discretized Gaussian error distributions $\chi$ with $B = 2^{-n^\epsilon} \cdot q \cdot \mathsf{poly}(n)$.

**LWE with Short Secrets.** In this work, we will be using a variant of the LWE problem called *LWE with Short Secrets*. In this variant, introduced by Applebaum et al. [ACPS09], the secret vector is also chosen from the noise distribution $\chi$. They showed that this variant is as hard as $\mathsf{LWE}$ for sufficiently large number of samples $m$.

**Assumption 2** (LWE with Short Secrets)**.** Let $n$, $m$ and $q$ be positive integers and $\chi$ a noise distribution on $\mathbb{Z}$. The LWE with Short Secrets assumption $(n, m, q, \chi)$-$\mathsf{LWE\text{-}ss}$, parameterized by $n, m, q, \chi$, states that the following distributions are computationally indistinguishable [8]:

$$\left\{(\mathbf{A}, \mathbf{S} \cdot \mathbf{A} + \mathbf{E}) \ : \ \begin{array}{c} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \\ \mathbf{S} \leftarrow \chi^{n \times n}, \mathbf{E} \leftarrow \chi^{n \times m} \end{array} \right\} \approx_c \left\{(\mathbf{A}, \mathbf{U}) \ : \ \begin{array}{c} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \\ \mathbf{U} \leftarrow \mathbb{Z}_q^{n \times m} \end{array} \right\}.$$

**Lattices with Trapdoors.** Lattices with trapdoors are lattices that are statistically indistinguishable from randomly chosen lattices, but have certain 'trapdoors' that allow efficient solutions to hard lattice problems.

**Definition 2.1** ([Ajt99, GPV08])**.** A trapdoor lattice sampler consists of algorithms $\mathsf{TrapGen}$ and $\mathsf{SamplePre}$ with the following syntax and properties:

- $\mathsf{TrapGen}(1^n, 1^m, q) \to (\mathbf{A}, T_\mathbf{A})$: The lattice generation algorithm is a randomized algorithm that takes as input the matrix dimensions $n, m$, modulus $q$, and outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a trapdoor $T_\mathbf{A}$.
- $\mathsf{SamplePre}(\mathbf{A}, T_\mathbf{A}, \mathbf{u}, \sigma) \to \mathbf{s}$: The presampling algorithm takes as input a matrix $\mathbf{A}$, trapdoor $T_\mathbf{A}$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$ and a parameter $\sigma \in \mathcal{R}$ (which determines the length of the output vectors). It outputs a vector $\mathbf{s} \in \mathbb{Z}_q^m$.

These algorithms must satisfy the following properties:

1. *Correct Presampling:* For all vectors $\mathbf{u}$, parameters $\sigma$, $(\mathbf{A}, T_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, and $\mathbf{s} \leftarrow \mathsf{SamplePre}(\mathbf{A}, T_\mathbf{A}, \mathbf{u}, \sigma)$, $\mathbf{A} \cdot \mathbf{s} = \mathbf{u}$ and $\|\mathbf{s}\|_\infty \leq \sqrt{m} \cdot \sigma$.
2. *Well Distributedness of Matrix:* The following distributions are statistically indistinguishable:
$$\{\mathbf{A} : (\mathbf{A}, T_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)\} \approx_s \{\mathbf{A} : \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}\}.$$
3. *Well Distributedness of Preimage:* For all $(\mathbf{A}, T_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, if $\sigma = \omega(\sqrt{n \cdot \log q \cdot \log m})$, then the following distributions are statistically indistinguishable:
$$\{\mathbf{s} : \mathbf{u} \leftarrow \mathbb{Z}_q^n, \mathbf{s} \leftarrow \mathsf{SamplePre}(\mathbf{A}, T_\mathbf{A}, \mathbf{u}, \sigma)\} \approx_s \mathcal{D}_{\mathbb{Z}^m, \sigma}.$$

These properties are satisfied by the gadget-based trapdoor lattice sampler of [MP12] for parameters $m$ such that $m = \Omega(n \cdot \log q)$.

[8] Applebaum et al. showed that $\{(\mathbf{A}, \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{s} \leftarrow \chi^n, \mathbf{e} \leftarrow \chi^m\} \approx_c \{(\mathbf{A}, \mathbf{u}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{u} \leftarrow \mathbb{Z}_q^m\}$, assuming LWE is hard. However, by a simple hybrid argument, we can replace vectors $\mathbf{s}, \mathbf{e}, \mathbf{u}$ with matrices $\mathbf{S}, \mathbf{E}, \mathbf{U}$ of appropriate dimensions.

## 2.2 Branching Programs

Branching programs are a model of computation used to capture space-bounded computations [BDFP86, Bar86]. In this work, we will be using a restricted notion called *permutation branching programs*.

**Definition 2.2** (Permutation Branching Program)**.** A permutation branching program of length $L$, width $w$ and input space $\{0,1\}^n$ consists of a sequence of $2L$ permutations $\sigma_{i,b} : [w] \to [w]$ for $1 \leq i \leq L, b \in \{0,1\}$, an input selection function $\mathsf{inp} : [L] \to [n]$, an accepting state $\mathsf{acc} \in [w]$ and a rejection state $\mathsf{rej} \in [w]$. The starting state $\mathsf{st}_0$ is set to be 1 without loss of generality. The branching program evaluation on input $x \in \{0,1\}^n$ proceeds as follows:

- For $i = 1$ to $L$,
  - Let $\mathsf{pos} = \mathsf{inp}(i)$ and $b = x_{\mathsf{pos}}$. Compute $\mathsf{st}_i = \sigma_{i,b}(\mathsf{st}_{i-1})$.
- If $\mathsf{st}_L = \mathsf{acc}$, output 1. If $\mathsf{st}_L = \mathsf{rej}$, output 0, else output $\perp$.

In a remarkable result, Barrington [Bar86] showed that any circuit of depth $d$ can be simulated by a permutation branching program of width 5 and length $4^d$.

**Theorem 2.2** ([Bar86])**.** For any boolean circuit $C$ with input space $\{0,1\}^n$ and depth $d$, there exists a permutation branching program $\mathsf{BP}$ of width 5 and length $4^d$ such that for all inputs $x \in \{0,1\}^n$, $C(x) = \mathsf{BP}(x)$.

This permutation property will be useful for proving security of our main construction in Section 4. We will also require that the permutation branching program has a fixed input-selector function $\mathsf{inp}$. In our construction, we will have multiple branching programs, and all of them must read the same input bit at any level $i \leq L$.

**Definition 2.3.** A permutation branching program with input space $\{0,1\}^n$ is said to have a fixed input-selector $\mathsf{inp}(\cdot)$ if for all $i \leq L$, $\mathsf{inp}(i) = i \bmod n$.

Any permutation branching program of length $L$ and input space $\{0,1\}^n$ can be easily transformed to a fixed input-selector branching program of length $n \cdot L$. In this work, we only require that all branching programs share the same input selector function $\mathsf{inp}(\cdot)$. The input selector which satisfies $\mathsf{inp}(i) = i \bmod n$ is just one possibility, and we stick with it for simplicity. We will use the following corollary, which follows from Theorem 2.2.

**Corollary 2.2.** For any boolean circuit $C$ with input space $\{0,1\}^n$ and depth $d$, there exists a *fixed-input selector* permutation branching program $\mathsf{BP}$ of width 5 and length $n \cdot 4^d$ such that for all inputs $x \in \{0,1\}^n$, $C(x) = \mathsf{BP}(x)$.

## 2.3 Public Key Encryption

A public key encryption (PKE) scheme $\mathcal{E}$ for message spaces $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ consists of the following polynomial-time algorithms.

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$. The setup algorithm takes as input the security parameter $\lambda$, and outputs a public-secret key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk}, m \in \mathcal{M}_\lambda) \to \mathsf{ct}$. The encryption algorithm takes as input a public key $\mathsf{pk}$ and a message $m$, and outputs a ciphertext $\mathsf{ct}$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m$. The decryption algorithm takes as input a secret key $\mathsf{sk}$ and a ciphertext $\mathsf{ct}$, and outputs a message $m$.

**Correctness.** A PKE scheme $\mathcal{E}$ for message spaces $\mathcal{M}$ is said to be correct if for all $\lambda$, $m \in \mathcal{M}_\lambda$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, we have that $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m$.

**Security.** The standard security notion for PKE schemes is IND-CPA security. Formally, it is defined as follows.

**Definition 2.4.** A public key encryption scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ is IND-CPA secure if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible functions $\mathrm{negl}(\cdot)$, such that the following function of $\lambda$ is bounded by $\mathrm{negl}(\cdot)$

$$\left|\Pr\left[\mathcal{A}(\mathsf{ct}) = b \middle| \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda); b \leftarrow \{0,1\} \\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk}); \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)\end{array}\right] - \frac{1}{2}\right|$$

## 2.4 Homomorphic Encryption

Homomorphic encryption [RAD78, Gen09] is a powerful extension of public key encryption that allows one to evaluate functions on ciphertexts. Homomorphic encryption schemes can be classified as either leveled or fully homomorphic encryption schemes. A leveled homomorphic encryption (LHE) scheme allows bounded depth computation over the ciphertexts. The setup algorithm takes as input a 'level bound' $\ell$ together with the security parameter, and outputs a public-secret key pair. Given an ciphertext $\mathsf{ct}$ corresponding to message $m$, one can use the evaluation algorithm to evaluate a bounded depth circuit $C$ on $\mathsf{ct}$, and the resulting ciphertext $\mathsf{ct}'$, when decrypted using the secret key, outputs $C(m)$ if the depth of $C$ is less than $\ell$. Fully homomorphic encryption, on the other hand, allows for arbitrary computation on the ciphertext.

### 2.4.1 Leveled Homomorphic Encryption

A secret key leveled homomorphic encryption scheme $\mathcal{HE}$ with message space $\{0,1\}$ consists of four algorithms $\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}$ with the following syntax:

1. $\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{sk}, \mathsf{ek})$ The setup algorithm takes as input the security parameter $\lambda$, bound on circuit depth $\ell$ and outputs a secret key $\mathsf{sk}$ and evaluation key $\mathsf{ek}$.
2. $\mathsf{Enc}(\mathsf{sk}, m \in \{0,1\}) \to \mathsf{ct}$ The encryption algorithm takes as input a secret key $\mathsf{sk}$, message $m \in \{0,1\}$ and outputs a ciphertext $\mathsf{ct}$.
3. $\mathsf{Eval}(\mathsf{ek}, C \in \mathcal{C}_\ell, \mathsf{ct}) \to \mathsf{ct}'$ The evaluation algorithm takes as input an evaluation key $\mathsf{ek}$, a circuit $C \in \mathcal{C}_\ell$, a ciphertext $\mathsf{ct}$ and outputs a ciphertext $\mathsf{ct}'$.
4. $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to x$ The decryption algorithm takes as input a secret key $\mathsf{sk}$ and ciphertext $\mathsf{ct}$ and outputs $x \in \{0,1\} \cup \{\perp\}$.

We will now define some properties of leveled homomorphic encryption schemes. Let $\mathcal{HE}$ be any homomorphic encryption scheme with message space $\{0,1\}$. First, we have the correctness property, which states that the decryption of a homomorphic evaluation on a ciphertext must be equal to the evaluation on the underlying message.

**Definition 2.5** (Correctness)**.** The scheme $\mathcal{HE}$ is said to be (perfectly) correct if for all security parameter $\lambda$, circuit-depth bound $\ell$, $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, circuit $C \in \mathcal{C}_\ell$ and message $m \in \{0,1\}$,

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{ek}, C, \mathsf{Enc}(\mathsf{sk}, m))) = C(m).$$

Next, we have the compactness property which requires that the size of the output of an evaluation on a ciphertext must not depend upon the evaluation circuit. In particular, we require that there exists one decryption circuit such that this circuit can decrypt any bounded-depth evaluations on ciphertexts.

**Definition 2.6** (Compactness)**.** A homomorphic encryption scheme $\mathcal{HE}$ is said to be compact if for all $\lambda$, $\ell$ there is a decryption circuit $C^{\mathsf{Dec}}_{\lambda,\ell}$ such that for all $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, $m \in \{0,1\}$, $C \in \mathcal{C}_\ell$, $C^{\mathsf{Dec}}_{\lambda,\ell}(\mathsf{sk}, \mathsf{Eval}(\mathsf{ek}, C, \mathsf{Enc}(\mathsf{sk}, m))) = C(m)$.

Finally, we require that the depth of the decryption circuit is bounded by a logarithmic function in the security parameter $\lambda$.

**Definition 2.7.** A compact homomorphic encryption scheme $\mathcal{HE}$ is said to have log-depth decryption circuit if for all $\lambda, \ell$, $\mathsf{depth}(C^{\mathsf{Dec}}_{\lambda,\ell}) = O(\log \lambda)$.

For security, we require that the underlying scheme is IND-CPA secure.

**Definition 2.8** (Security)**.** A homomorphic encryption scheme $\mathcal{HE}$ is secure if $\Gamma = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ is IND-CPA secure (Definition 2.4).

Starting with the work of Gentry [Gen09], there has been a long line of interesting works seeking to improve the efficiency/security of homomorphic encryption schemes. Today, we have LHE schemes [BV11, BGV12, GSW13] with log-depth decryption circuits that can be proven secure under the Learning with Errors assumption.

### 2.4.2 Fully Homomorphic Encryption

Fully homomorphic encryption is a stronger notion of homomorphic encryption where the evaluator can perform unbounded computations on the ciphertext. As a result, the algorithm does not need to take the depth bound as input.

A secret key fully homomorphic encryption scheme $\mathcal{HE}$ with message space $\{0,1\}$ consists of four algorithms $\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}$ with the following syntax:

1. $\mathsf{Setup}(1^\lambda) \to (\mathsf{sk}, \mathsf{ek})$ The setup algorithm takes as input the security parameter $\lambda$ and outputs a secret key $\mathsf{sk}$ and evaluation key $\mathsf{ek}$.
2. $\mathsf{Enc}(\mathsf{sk}, m \in \{0,1\}) \to \mathsf{ct}$ The encryption algorithm takes as input a secret key $\mathsf{sk}$, message $m \in \{0,1\}$ and outputs a ciphertext $\mathsf{ct}$.
3. $\mathsf{Eval}(\mathsf{ek}, C, \mathsf{ct}) \to \mathsf{ct}'$ The evaluation algorithm takes as input an evaluation key $\mathsf{ek}$, a circuit $C$, a ciphertext $\mathsf{ct}$ and outputs a ciphertext $\mathsf{ct}'$.
4. $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to x$ The decryption algorithm takes as input a secret key $\mathsf{sk}$ and ciphertext $\mathsf{ct}$ and outputs $x \in \{0,1\} \cup \{\perp\}$.

**Definition 2.9** (Correctness)**.** The scheme $\mathcal{HE}$ is said to be (perfectly) correct if for all security parameter $\lambda$, $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, circuit $C$ and message $m \in \{0,1\}$,

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{ek}, C, \mathsf{Enc}(\mathsf{sk}, m))) = C(m).$$

The security and compactness requirement is same as that for leveled homomorphic encryption.

## 2.5 Pairwise Independent Hash Functions

**Definition 2.10.** Let $\mathcal{H}$ be a family of hash functions where each $h \in \mathcal{H}$ has domain $\{0,1\}^\ell$ and co-domain $\{0,1\}^{\ell_\mathcal{H}}$. The family $\mathcal{H}$ is said to be a pairwise independent hash function family if for all $x, y \in \{0,1\}^\ell$ and $a, b \in \{0,1\}^{\ell_\mathcal{H}}$ s.t. $x \neq y$,

$$\Pr_{h \leftarrow \mathcal{H}}[h(x) = a \ \wedge \ h(y) = b] = \frac{1}{\left(2^{\ell_\mathcal{H}}\right)^2}.$$

A simple family of pairwise independent hash functions is the following set of functions:

$$\{h_{\mathbf{A},\mathbf{b}}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b} \ : \ \mathbf{A} \in \mathbb{Z}_2^{\ell_\mathcal{H} \times \ell}, \mathbf{b} \in \mathbb{Z}_2^{\ell_\mathcal{H}}\}.$$

This family can be implemented using log-depth circuits.

## 2.6 $\mathbf{NC}^1$ Pseudorandom Generators with Polynomial Stretch

In this work, we require polynomial stretch pseudorandom generators which can be implemented by polynomial length branching programs.

**Definition 2.11.** Let $\ell, \ell_{\mathrm{PRG}}$ be polynomials. A circuit class $\mathrm{PRG}_{\ell,\ell_{\mathrm{PRG}}} = \{\mathrm{PRG}_\lambda\}_\lambda$ where $\mathrm{PRG}_\lambda : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{\ell_{\mathrm{PRG}}(\lambda)}$, is said to be a pseudorandom generator if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that the following function of $\lambda$ is bounded by $\mathrm{negl}(\cdot)$:

$$\left|\Pr\left[1 \leftarrow \mathcal{A}(\mathrm{PRG}_\lambda(s)) : s \leftarrow \{0,1\}^{\ell(\lambda)}\right] - \Pr\left[1 \leftarrow \mathcal{A}(u) : u \leftarrow \{0,1\}^{\ell_{\mathrm{PRG}}(\lambda)}\right]\right|$$

Using PRFs in $\mathbf{NC}^1$, we can show that there exist PRGs of logarithmic depth which can achieve any polynomial stretch. Low depth PRFs, in turn, can be constructed from the Learning with Errors assumption.

**Theorem 2.3** (PRFs in $\mathbf{NC}^1$ [BPR12])**.** For some $\sigma > 0$, suitable universal constant $C > 0$, modulus $p \geq 2$, any $m = \mathsf{poly}(n)$, let $\chi = \mathcal{D}_{\mathbb{Z},\sigma}$ and $q \geq p \cdot k(C\sigma\sqrt{n})^k \cdot n^{\omega(1)}$, assuming hardness of $(n, m, q, \chi)$-$\mathsf{LWE}$, there exists a function family PRF consisting of functions from $\{0,1\}^k$ to $\mathbb{Z}_p^{m\times n}$ that satisfies pseudorandomness property and the entire function can be computed in $\mathbf{TC}^0 \subseteq \mathbf{NC}^1$.

From Theorems 2.2 and 2.3, the following corollary is immediate.

**Corollary 2.3.** There exists a polynomial $p$ such that for any polynomial $\ell_{\mathrm{PRG}}$, there exists a family of pseudorandom generators $\{\mathrm{PRG}_\lambda\}$ where $\mathrm{PRG}_\lambda : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{\ell_{\mathrm{PRG}}(\lambda)}$, and for all $i \in [\ell_{\mathrm{PRG}}(\lambda)]$, the $i^{th}$ bit of $\mathrm{PRG}_\lambda$ can be computed by a branching program of length $p(\lambda)$. This family of pseudorandom generators can be proven secure assuming hardness of $(n, m, q, \chi)$-$\mathsf{LWE}$ with parameters as in Theorem 2.3.

# 3 Lockable Obfuscation

In this section, we define the notion of lockable obfuscation, and discuss how it is related to other cryptographic primitives. An lockable obfuscator takes as input a program $P$, a message $\mathsf{msg}$, and a 'lock' $\alpha$. It outputs an obfuscated program $\widetilde{P}$ which has the same domain as the program $P$. The program $\widetilde{P}$, on input $x$, outputs the message $\mathsf{msg}$ if $P(x) = \alpha$. If not, then with overwhelming probability, it will output $\perp$. For security, we require that for all programs $P$ and messages $\mathsf{msg}$, $\widetilde{P}$ for a uniformly random $\alpha$ can be efficiently simulated given only the sizes of $P$ and $\mathsf{msg}$. We will now present the syntax, correctness and security definition.

Let $n, m, d$ be polynomials, and $\mathcal{C}_{n,m,d}(\lambda)$ be the class of depth $d(\lambda)$ circuits with $n(\lambda)$ bit input and $m(\lambda)$ bit output. A lockable obfuscator for $\mathcal{C}_{n,m,d}$ consists of algorithms $\mathsf{Obf}$ and $\mathsf{Eval}$ with the following syntax. Let $\mathcal{M}$ be the message space.

- $\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha) \to \widetilde{P}$. The obfuscation algorithm is a randomized algorithm that takes as input the security parameter $\lambda$, a program $P \in \mathcal{C}_{n,m,d}$, message $\mathsf{msg} \in \mathcal{M}$ and 'lock string' $\alpha \in \{0,1\}^{m(\lambda)}$. It outputs a program $\widetilde{P}$.
- $\mathsf{Eval}(\widetilde{P}, x) \to y \in \mathcal{M} \cup \{\perp\}$. The evaluator is a deterministic algorithm that takes as input a program $\widetilde{P}$ and a string $x \in \{0,1\}^{n(\lambda)}$. It outputs $y \in \mathcal{M} \cup \{\perp\}$.

## 3.1 Correctness

For correctness, we require that if $P(x) = \alpha$, then the obfuscated program $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha)$, evaluated on input $x$, outputs $\mathsf{msg}$, and if $P(x) \neq \alpha$, then $\widetilde{P}$ outputs $\perp$ on input $x$. Here, we give three correctness definitions, which differ in the case where $P(x) \neq \alpha$.

**Definition 3.1** (Perfect Correctness)**.** Let $n, m, d$ be polynomials. A lockable obfuscation scheme for $C_{n,m,d}$ and message space $\mathcal{M}$ is said to be perfectly correct if it satisfies the following properties:

1. For all security parameters $\lambda$, inputs $x \in \{0,1\}^{n(\lambda)}$, programs $P \in \mathcal{C}_{n,m,d}$ and messages $\mathsf{msg} \in \mathcal{M}$, if $P(x) = \alpha$, then
$$\mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha)), x) = \mathsf{msg}.$$

2. For all security parameters $\lambda$, inputs $x \in \{0,1\}^{n(\lambda)}$, programs $P \in \mathcal{C}_{n,m,d}$ and messages $\mathsf{msg} \in \mathcal{M}$, if $P(x) \neq \alpha$, then
$$\mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha)), x) = \perp .$$

**Definition 3.2** (Statistical Correctness)**.** An lockable obfuscation scheme is said to be statistically correct if it satisfies the following properties:

1. For all security parameters $\lambda$, inputs $x \in \{0,1\}^{n(\lambda)}$, programs $P \in \mathcal{C}_{n,m,d}$ and messages $\mathsf{msg} \in \mathcal{M}$, if $P(x) = \alpha$, then
$$\mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha)), x) = \mathsf{msg}.$$

2. For all security parameters $\lambda$, programs $P \in \mathcal{C}_{n,m,d}$, $\alpha \in \{0,1\}^{m(\lambda)}$ and $\mathsf{msg} \in \mathcal{M}$,
$$\Pr[\exists\ x \ \text{ s.t. } P(x) \neq \alpha \text{ and } \mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha), x) = \mathsf{msg}] \leq \mathrm{negl}(\lambda)$$
where the probability is taken over the random coins used during obfuscation.

**Definition 3.3** (Semi-statistical Correctness)**.** Let $n, m, d$ be polynomials. A lockable obfuscation scheme for $C_{n,m,d}$ and message space $\mathcal{M}$ is said to be semi-statistical correct if it satisfies the following properties:

1. For all security parameters $\lambda$, inputs $x \in \{0,1\}^{n(\lambda)}$, programs $P \in \mathcal{C}_{n,m,d}$ and messages $\mathsf{msg} \in \mathcal{M}$, if $P(x) = \alpha$, then
$$\mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha)), x) = \mathsf{msg}.$$

2. For all security parameters $\lambda$, inputs $x \in \{0,1\}^{n(\lambda)}$, programs $P \in \mathcal{C}_{n,m,d}$, $\alpha \in \{0,1\}^{m(\lambda)}$ such that $P(x) \neq \alpha$ and $\mathsf{msg} \in \mathcal{M}$,
$$\Pr[\mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha), x) = \mathsf{msg}] \leq \mathrm{negl}(\lambda)$$
where the probability is taken over the random coins used during obfuscation.

Looking ahead, our construction satisfies the semi-statistical correctness condition, and will be sufficient for our applications.

## 3.2 Security

**Definition 3.4.** Let $n, m, d$ be polynomials. A lockable obfuscation scheme $(\mathsf{Obf}, \mathsf{Eval})$ for $\mathcal{C}_{n,m,d}$ and message space $\mathcal{M}$ is said to be secure if there exists a PPT simulator $\mathsf{Sim}$ such that for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that the following function is bounded by $\mathrm{negl}(\cdot)$:
$$\left| \Pr\left[ \mathcal{A}_1(\tilde{P}_b, \mathsf{st}) = b : \begin{array}{c} (P \in \mathcal{C}_{n,m,d}, \mathsf{msg} \in \mathcal{M}, \mathsf{st}) \leftarrow \mathcal{A}_0(1^\lambda) \\ b \leftarrow \{0,1\}, \alpha \leftarrow \{0,1\}^{m(\lambda)} \\ \tilde{P}_0 \leftarrow \mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha) \\ \tilde{P}_1 \leftarrow \mathsf{Sim}(1^\lambda, 1^{|P|}, 1^{|\mathsf{msg}|}) \end{array} \right] - \frac{1}{2} \right|$$

# 4 Our Construction

In this section, we present our lockable obfuscation scheme. For any polynomials $\ell_{\text{in}}, \ell_{\text{out}}, d$ such that $\ell_{\text{out}} = \omega(\log \lambda)$, we construct a lockable obfuscation scheme $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ for the circuit class $\mathcal{C}_{\ell_{\text{in}},\ell_{\text{out}},d}$. The message space for our construction will be $\{0,1\}$, although one can trivially extend it to $\{0,1\}^{\ell(\lambda)}$ for any polynomial $\ell$. More details are provided in Appendix C.

The tools required for our construction are as follows:

- A compact leveled homomorphic bit encryption scheme $(\mathsf{LHE.Setup}, \mathsf{LHE.Enc}, \mathsf{LHE.Eval}, \mathsf{LHE.Dec})$ with decryption circuit of depth $d_{\mathsf{Dec}}(\lambda)$ and ciphertexts of length $\ell_{\mathsf{ct}}(\lambda)$.
- A pairwise independent hash family $\mathcal{H} : \{0,1\}^{\ell_{\text{out}}(\lambda)} \to \{0,1\}^{\ell_{\mathcal{H}}(\lambda)}$ which can be implemented by a circuit of logarithmic depth $d_{\mathcal{H}}(\lambda)$.
- A pseudorandom generator $\text{PRG} : \{0,1\}^{\ell_{\mathcal{H}}(\lambda)} \to \{0,1\}^{\ell_{\text{PRG}}(\lambda)}$ of depth $d_{\text{PRG}}(\lambda)$.

For notational convenience, let $\ell_{\text{in}} = \ell_{\text{in}}(\lambda)$, $\ell_{\text{out}} = \ell_{\text{out}}(\lambda)$, $\ell_{\mathcal{H}} = \ell_{\mathcal{H}}(\lambda)$, $\ell_{\text{PRG}} = \ell_{\text{PRG}}(\lambda)$, $d_{\mathsf{Dec}} = d_{\mathsf{Dec}}(\lambda)$, $d_{\mathcal{H}} = d_{\mathcal{H}}(\lambda)$, $d_{\text{PRG}} = d_{\text{PRG}}(\lambda)$ and $d = d(\lambda)$.

Fix any $\epsilon < 1/2$. Let $\chi$ be a $B$-bounded discrete Gaussian distribution with parameter $\sigma$ such that $B = \sqrt{m} \cdot \sigma$. Let $n, m, \ell, \sigma, q, \mathsf{Bd}$ be parameters with the following constraints:

- $n = \mathsf{poly}(\lambda)$ and $q \leq 2^{n^\epsilon}$ (for LWE security)
- $\ell_{\mathcal{H}} = \omega(\log \lambda)$ (for PRG security)
- $\ell_{\text{out}} - \ell_{\mathcal{H}} = \omega(\log \lambda)$ (for strong extraction)
- $m = \Omega(n \cdot \log q)$ (for $\mathsf{SamplePre}$)
- $\sigma = \omega(\sqrt{n \cdot \log q \cdot \log m})$ (for Preimage Well Distributedness)
- $\ell_{\text{PRG}} = n \cdot m \cdot \log q + \omega(\log n)$ (for applying Leftover Hash Lemma)
- $\ell_{\text{PRG}} - \ell_{\mathcal{H}} = \omega(\log \lambda)$ (for correctness of scheme)
- $\mathsf{Bd} = \ell_{\text{PRG}} \cdot L \cdot (m^2 \cdot \sigma)^L < q^{1/4}$ (for correctness of scheme)
  (where $L = \ell_{\text{out}} \cdot \ell_{\mathsf{ct}} \cdot 4^{d_{\mathsf{Dec}} + d_{\mathcal{H}} + d_{\text{PRG}}}$)

First, note that it is important that $L = \lambda^c$ for some constant $c$ and $\mathsf{Bd} = \ell_{\text{PRG}} \cdot L \cdot (m \cdot \sigma)^L < q^{1/4}$. This crucially relies on the fact that the LHE scheme is compact (so that $\ell_{\mathsf{ct}}$ is bounded by a polynomial) and that the LHE decryption, hash function evaluation and PRG computation can be performed by a log depth circuit (i.e, have poly length branching programs). The constant $c$ depends on the LHE scheme, pairwise independent hash function and PRG.

One possible setting of parameters is as follows: $n = \lambda^{4c/\epsilon}$, $m = n^{1+\epsilon}$, $q = 2^{n^\epsilon}$, $\sigma = n$, $\ell_{\text{PRG}} = n^{2\epsilon+2}$ and $\ell_{\mathcal{H}} = \ell_{\text{out}}/2$.

We will now describe the obfuscation and evaluation algorithms.

- $\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha)$: The obfuscation algorithm takes as input a program $P \in \mathcal{C}_{\ell_{\text{in}},\ell_{\text{out}},d}$, message $\mathsf{msg} \in \{0,1\}$ and $\alpha \in \{0,1\}^{\ell_{\text{out}}}$. The obfuscator proceeds as follows:
  1. First, it chooses the LHE key pair as $(\mathsf{lhe.sk}, \mathsf{lhe.ek}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^{d \log d})$.[9]
  2. Next, it encrypts the program $P$. It sets $\mathbf{ct} \leftarrow \mathsf{LHE.Enc}(\mathsf{lhe.sk}, P)$.[10]
  3. It then chooses a pairwise independent hash function $h \leftarrow \mathcal{H}$. Let $\beta = \text{PRG}(h(\alpha))$.

[9] We set the LHE depth bound to be $d \log d$, where the extra log factor is to account for the constant blowup involved in using a universal circuit. In particular, we can set the LHE depth bound to be $c \cdot d$ where $c$ is some fixed constant depending on the universal circuit.

[10] Note that LHE scheme supports bit encryption. Therefore, to encrypt $P$, a multi-bit message, the $\mathsf{FHE.Enc}$ algorithm will be run independently on each bit of $P$. However, for notational convenience throughout this section we overload the notation and use $\mathsf{FHE.Enc}$ and $\mathsf{FHE.Dec}$ algorithms to encrypt and decrypt multi-bit messages respectively.

4. Next, consider the following circuit $Q$ which takes as input $\ell_{\text{out}} \cdot \ell_{\mathsf{ct}}$ bits of input and outputs $\ell_{\text{PRG}}$ bits. $Q$ takes as input $\ell_{\text{out}}$ LHE ciphertexts $\{\mathsf{ct}_i\}_{i \le \ell_{\text{out}}}$, has LHE secret key $\mathsf{lhe.sk}$ hardwired and computes the following — (1) it decrypts each input ciphertext $\mathsf{ct}_i$ (in parallel) to get string $x$ of length $\ell_{\text{out}}$ bits, (2) it sequentially applies the hash function and PRG on $x$ and outputs $\text{PRG}(h(x))$. Concretely, $Q(\mathsf{ct}_1, \ldots, \mathsf{ct}_{\ell_{\text{out}}}) = \text{PRG}\big(h\big(\mathsf{LHE.Dec}(\mathsf{lhe.sk}, \mathsf{ct}_1) \,||\, \cdots \,||\, \mathsf{LHE.Dec}(\mathsf{lhe.sk}, \mathsf{ct}_{\ell_{\text{out}}})\big)\big)$.

   For $i \le \ell_{\text{PRG}}$, we use $\mathsf{BP}^{(i)}$ to denote the fixed-input selector permutation branching program that outputs the $i^{th}$ bit of output of circuit $Q$. Note that $Q$ has depth $d_{\mathsf{tot}} = d_{\mathsf{Dec}} + d_{\mathcal{H}} + d_{\text{PRG}}$. By Corollary 2.2, we know that each branching program $\mathsf{BP}^{(i)}$ has length $L = \ell_{\text{out}} \cdot \ell_{\mathsf{ct}} \cdot 4^{d_{\mathsf{tot}}}$ and width 5.
5. Finally, the obfuscator creates matrix components which enable the evaluator to compute $\mathsf{msg}$ if it has an input strings (ciphertexts) $\mathsf{ct}_1, \ldots, \mathsf{ct}_{\ell_{\text{out}}}$ such that $Q(\mathsf{ct}_1, \ldots, \mathsf{ct}_{\ell_{\text{out}}}) = \beta$. Concretely, it runs the (randomized) routine $\mathsf{Comp\text{-}Gen}$ (defined in Figure 1). This routine takes as input the circuit $Q$ in the form of $\ell_{\text{PRG}}$ branching programs $\{\mathsf{BP}^{(i)}\}_i$, string $\beta$ and message $\mathsf{msg}$. Let $\left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, \left\{\mathbf{C}_j^{(i,0)}, \mathbf{C}_j^{(i,1)}\right\}_{i,j}\right) \leftarrow \mathsf{Comp\text{-}Gen}(\{\mathsf{BP}^{(i)}\}_i, \beta, \mathsf{msg})$.
6. The final obfuscated program consists of the LHE evaluation key $\mathsf{ek} = \mathsf{lhe.ek}$, LHE ciphertexts $\mathbf{ct}$, together with the components $\left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, \left\{(\mathbf{C}_j^{(i,0)}, \mathbf{C}_j^{(i,1)})\right\}_{i,j}\right)$.

- $\mathsf{Eval}(\tilde{P}, x)$: The evaluation algorithm takes as input $\tilde{P} = \left(\mathsf{ek}, \mathbf{ct}, \left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, \left\{(\mathbf{C}_j^{(i,0)}, \mathbf{C}_j^{(i,1)})\right\}_{i,j}\right)$ and input $x \in \{0,1\}^{\ell_{\text{in}}}$. It performs the following steps.

  1. The evaluator first constructs a universal circuit $U_x(\cdot)$ with $x$ hardwired as input. This universal circuit takes a circuit $C$ as input and outputs $U_x(C) = C(x)$. Using the universal circuit of Cook and Hoover [CH85], it follows that $U_x(\cdot)$ has depth $O(d)$.
  2. Next, it performs homomorphic evaluation on $\mathbf{ct}$ using circuit $U_x(\cdot)$. It computes $\widetilde{\mathbf{ct}} = \mathsf{LHE.Eval}(\mathsf{ek}, U_x(\cdot), \mathbf{ct})$. Note that $\ell_{\mathsf{ct}} \cdot \ell_{\text{out}}$ denotes the length of $\widetilde{\mathbf{ct}}$ (as a bitstring), and let $\widetilde{\mathbf{ct}}_i$ denote the $i^{th}$ bit of $\widetilde{\mathbf{ct}}$.
  3. The evaluator then obliviously evaluates the $\ell_{\text{PRG}}$ branching programs on input $\widetilde{\mathbf{ct}}$ using the matrix components. It calls the component evaluation algorithm $\mathsf{Comp\text{-}Eval}$ (defined in Figure 2). Let $y = \mathsf{Comp\text{-}Eval}\left(\widetilde{\mathbf{ct}}, \left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, \left\{(\mathbf{C}_j^{(i,0)}, \mathbf{C}_j^{(i,1)})\right\}_{i,j}\right)\right)$. The evaluator outputs $y$.

## 4.1 Correctness

We will prove that the lockable obfuscation scheme described above satisfies the semi-statistical correctness property (see Definition 3.3). To prove this, we need to prove that if $P(x) = \alpha$, then the evaluation algorithm always outputs the message, and if $P(x) \neq \alpha$, then with high probability, it outputs $\perp$.

First, we will prove the following lemma about the $\mathsf{Comp\text{-}Gen}$ and $\mathsf{Comp\text{-}Eval}$ routines. Intuitively, this lemma states that for all fixed input branching programs $\{\mathsf{BP}^{(i)}\}_i$, strings $\beta$ and messages $\mathsf{msg}$, if $\mathsf{BP}^{(i)}(z) = \beta_i$ for all $i \le \ell_{\text{PRG}}$, then the component evaluator outputs $\mathsf{msg}$.

**Lemma 4.1.** For any set of branching programs $\{\mathsf{BP}^{(i)}\}_{i \le \ell_{\text{PRG}}}$, string $\beta \in \{0,1\}^{\ell_{\text{PRG}}}$, message $\mathsf{msg} \in \{0,1\}$ and input $z$,

1. if $\mathsf{BP}^{(i)}(z) = \beta_i$ for all $i \le \ell_{\text{PRG}}$, then $\mathsf{Comp\text{-}Eval}(z, \mathsf{Comp\text{-}Gen}(\{\mathsf{BP}^{(i)}\}_i, \beta, \mathsf{msg})) = \mathsf{msg}$.
2. if $\mathsf{BP}^{(i)}(z) \neq \beta_i$ for some $i \le \ell_{\text{PRG}}$, then

$$\Pr[\mathsf{Comp\text{-}Eval}(z, \mathsf{Comp\text{-}Gen}(\{\mathsf{BP}^{(i)}\}_i, \beta, \mathsf{msg})) = \perp] \ge 1 - \text{negl}(\lambda).$$

Comp-Gen

**Input:** $\{\mathsf{BP}^{(i)}\}_i$, $\beta \in \{0,1\}^{\ell_{\mathrm{PRG}}}$, $\mathsf{msg} \in \{0,1\}$

**Output:** Components $\left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, \{(\mathbf{C}_{\mathsf{level}}^{(i,0)}, \mathbf{C}_{\mathsf{level}}^{(i,1)})\}_{i \le \ell_{\mathrm{PRG}}, \mathsf{level} \le L}\right)$.

(a) Let $\mathsf{BP}^{(i)} = \left(\left\{\sigma_{j,b}^{(i)} : [5] \to [5]\right\}_{j \in [L], b \in \{0,1\}}, \mathsf{acc}^{(i)} \in [5], \mathsf{rej}^{(i)} \in [5]\right)$ for all $i \le \ell_{\mathrm{PRG}}$.

(b) First, it chooses a matrix for each state of each branching program. Recall, there are $\ell_{\mathrm{PRG}}$ branching programs, and each branching program has $L$ levels, and each level has 5 states. For each $i \le \ell_{\mathrm{PRG}}$, $j \in [0, L-1]$, it chooses a matrix of dimensions $5n \times m$ along with its trapdoors (independently) as $(\mathbf{B}_j^{(i)}, T_j^{(i)}) \leftarrow \mathsf{TrapGen}(1^{5n}, 1^m, q)$. The matrix $\mathbf{B}_j^{(i)}$ can be parsed as follows

$$\mathbf{B}_j^{(i)} = \begin{bmatrix} \mathbf{B}_{j,1}^{(i)} \\ \vdots \\ \mathbf{B}_{j,5}^{(i)} \end{bmatrix}$$

where matrices $\mathbf{B}_{j,k}^{(i)} \in \mathbb{Z}_q^{n \times m}$ for $k \le 5$. The matrix $\mathbf{B}_{j,k}^{(i)}$ corresponds to state $k$ at level $j$ of branching program $\mathsf{BP}^{(i)}$.

(c) For the top level, it chooses top level matrices $\mathbf{B}_{L,k}^{(i)}$ for each $i \le \ell_{\mathrm{PRG}}, k \le 5$, uniformly at random, subject to the following constraint:

$$\sum_{i \,:\, \beta_i = 0} \mathbf{B}_{L,\mathsf{rej}^{(i)}}^{(i)} + \sum_{i \,:\, \beta_i = 1} \mathbf{B}_{L,\mathsf{acc}^{(i)}}^{(i)} = \mathbf{0}^{n \times m} \text{ if } \mathsf{msg} = 0.$$

$$\sum_{i \,:\, \beta_i = 0} \mathbf{B}_{L,\mathsf{rej}^{(i)}}^{(i)} + \sum_{i \,:\, \beta_i = 1} \mathbf{B}_{L,\mathsf{acc}^{(i)}}^{(i)} = \sqrt{q} \cdot \left[\mathbf{I}_n \,||\, \mathbf{0}^{n \times (m-n)}\right] \text{ if } \mathsf{msg} = 1.$$

(d) Next, it generates the components for each level. For each level $\mathsf{level} \in [1, L]$, do the following:

i. Choose matrices $\mathbf{S}_{\mathsf{level}}^{(0)}, \mathbf{S}_{\mathsf{level}}^{(1)} \leftarrow \chi^{n \times n}$ and $\mathbf{E}_{\mathsf{level}}^{(i,0)}, \mathbf{E}_{\mathsf{level}}^{(i,1)} \leftarrow \chi^{5n \times m}$ for $i \le \ell_{\mathrm{PRG}}$. If either $\mathbf{S}_{\mathsf{level}}^{(0)}$ or $\mathbf{S}_{\mathsf{level}}^{(1)}$ has determinant zero, then set it to be $\mathbf{I}_n$.

ii. For $b \in \{0,1\}$, set matrix $\mathbf{D}_{\mathsf{level}}^{(i,b)}$ as a permutation of the matrix blocks of $\mathbf{B}_{\mathsf{level}}^{(i)}$ according to the permutation $\sigma_{\mathsf{level},b}^{(i)}(\cdot)$. More formally, for $i \le \ell_{\mathrm{PRG}}$, set

$$\mathbf{D}_{\mathsf{level}}^{(i,b)} = \begin{bmatrix} \mathbf{B}_{\mathsf{level},\sigma_{\mathsf{level},b}^{(i)}(1)}^{(i)} \\ \vdots \\ \mathbf{B}_{\mathsf{level},\sigma_{\mathsf{level},b}^{(i)}(5)}^{(i)} \end{bmatrix}.$$

iii. Set $\mathbf{M}_{\mathsf{level}}^{(i,b)} = \left(\mathbf{I}_5 \otimes \mathbf{S}_{\mathsf{level}}^{(b)}\right) \cdot \mathbf{D}_{\mathsf{level}}^{(i,b)} + \mathbf{E}_{\mathsf{level}}^{(i,b)}$ for $i \le \ell_{\mathrm{PRG}}$.

iv. Compute $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{\mathsf{level}-1}^{(i)}, T_{\mathsf{level}-1}^{(i)}, \sigma, \mathbf{M}_{\mathsf{level}}^{(i,b)})$

(e) Output $\left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, \{(\mathbf{C}_{\mathsf{level}}^{(i,0)}, \mathbf{C}_{\mathsf{level}}^{(i,1)})\}_{i \le \ell_{\mathrm{PRG}}, \mathsf{level} \le L}\right)$.

Figure 1: Routine Comp-Gen

*Proof.* The proof of this lemma will be similar to the proof of correctness of the testing algorithm in [GKW17b, Section 4.2].

Recall that the component generation algorithm chooses matrices $\mathbf{B}_j^{(i)}$ for each $i \le \ell_{\mathrm{PRG}}$, $j \le L$, $\mathbf{S}_j^{(0)}, \mathbf{S}_j^{(1)}$ for each $j \le L$ and $\mathbf{E}_j^{(i,0)}, \mathbf{E}_j^{(i,1)}$ for each $i \le \ell_{\mathrm{PRG}}, j \le L$. Note that the $\mathbf{S}_j^{(b)}$ and $\mathbf{E}_j^{(i,b)}$ matrices have $l_\infty$ norm bounded by $\sigma \cdot m$ since they are chosen from truncated Gaussian distribution with parameter $\sigma$.

We start by introducing some notations for this proof.

- $\mathsf{st}_j^{(i)}$ : the state of $\mathsf{BP}^{(i)}$ after $j$ steps when evaluated on $z$

Comp-Eval

**Input:** Input string $z$, Components $\left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, \{(\mathbf{C}_{\mathsf{level}}^{(i,0)}, \mathbf{C}_{\mathsf{level}}^{(i,1)})\}_{i\leq \ell_{\mathrm{PRG}}, \mathsf{level}\leq L}\right)$.
**Output:** $y \in \{0, 1, \perp\}$.

(a) For each $i \in [1, \ell_{\mathrm{PRG}}]$, do the following

i. Set $\mathbf{M}^{(i)} = \mathbf{B}_{0,1}^{(i)}$.

ii. For $j = 1$ to $L$, do the following

- If $z_{\mathsf{inp}(j)} = 0$, set $\mathbf{M}^{(i)} = \mathbf{M}^{(i)} \cdot \mathbf{C}_j^{(i,0)}$. Else, set $\mathbf{M}^{(i)} = \mathbf{M}^{(i)} \cdot \mathbf{C}_j^{(i,1)}$.

(b) Compute $\mathbf{M} = \sum_i \mathbf{M}^{(i)}$. Let $\mathbf{M} = \left[\mathbf{M}^{(1)} \,||\, \mathbf{M}^{(2)}\right]$ where $\mathbf{M}^{(1)} \in \mathbb{Z}_q^{n\times n}$ (i.e. a square matrix). Next, do the following

- If $\|\mathbf{M}\|_\infty \leq \mathsf{Bd}$, output 0.
- Otherwise, if $\left\|\mathbf{M}^{(2)}\right\|_\infty \leq \mathsf{Bd}$ and $\|\mathbf{M}\|_\infty \in \left[\sqrt{q} - \mathsf{Bd}, (\sqrt{q}+1)\cdot \mathsf{Bd}\right]$, output 1.
- Else output $\perp$.

Figure 2: Routine Comp-Eval

- $\mathbf{S}_j = \mathbf{S}_j^{(z_{\mathsf{inp}(j)})}, \quad \mathbf{E}_j^{(i)} = \mathbf{E}_j^{(i, z_{\mathsf{inp}(j)})}, \quad \mathbf{C}_j^{(i)} = \mathbf{C}_j^{(i, z_{\mathsf{inp}(j)})}$ for all $j \leq L$
- $\mathbf{\Gamma}_{j^*} = \prod_{j=1}^{j^*} \mathbf{S}_j$ for all $j^* \leq L$
- $\mathbf{\Delta}_{j^*}^{(i)} = \mathbf{B}_{0,1}^{(i)} \cdot \left(\prod_{j=1}^{j^*} \mathbf{C}_j^{(i)}\right), \quad \widetilde{\mathbf{\Delta}}_{j^*}^{(i)} = \mathbf{\Gamma}_{j^*} \cdot \mathbf{B}_{j^*, \mathsf{st}_{j^*}^{(i)}}^{(i)}, \quad \mathbf{Err}_{j^*}^{(i)} = \mathbf{\Delta}_{j^*}^{(i)} - \widetilde{\mathbf{\Delta}}_{j^*}^{(i)}$ for all $j^* \leq L$

Observe that the Comp-Eval algorithm computes matrix $\mathbf{M} = \sum_{i=1}^{\ell_{\mathrm{PRG}}} \mathbf{\Delta}_L^{(i)}$. First, we show that for all $i \leq \ell_{\mathrm{PRG}}$, $j^* \leq L$, $\mathbf{Err}_{j^*}^{(i)}$ is small and bounded. This would help us in arguing that matrices $\sum_{i=1}^{\ell_{\mathrm{PRG}}} \mathbf{\Delta}_L^{(i)}$ and $\sum_{i=1}^{\ell_{\mathrm{PRG}}} \widetilde{\mathbf{\Delta}}_L^{(i)}$ are very close to each other.

Now, since we already know that $\sum_{i=1}^{\ell_{\mathrm{PRG}}} \widetilde{\mathbf{\Delta}}_L^{(i)}$ is close to all zeros matrix if $\mathsf{msg} = 0$ and is close to $\sqrt{q} \cdot \left[\mathbf{\Gamma}_L \,||\, \mathbf{0}^{n\times(m-n)}\right]$ if $\mathsf{msg} = 1$, therefore combining this with the fact that $\mathbf{Err}_{j^*}^{(i)}$ is small and bounded, we can conclude by arguing that matrix $\mathbf{M}$ will have appropriately bounded entries dependending on the value of $\mathsf{msg}$.

First, we show that $\mathbf{Err}_{j^*}^{(i)}$ is bounded.

**Claim 4.1.** $\forall\, i \in \{1, \ldots, \ell_{\mathrm{PRG}}\}, j^* \in \{1, \ldots, L\}, \quad \left\|\mathbf{Err}_{j^*}^{(i)}\right\|_\infty \leq j^* \cdot \left(m^2 \cdot \sigma\right)^{j^*}$.

*Proof.* The above claim is proven by induction over $j^*$, and all arguments hold irrespective of the value of $i$. Therefore, for simplicity of notation, we will drop the dependence on $i$.

**Base case** ($j^* = 1$)**.** We know that $\mathbf{\Delta}_1 = \mathbf{B}_{0,1} \cdot \mathbf{C}_1 = \mathbf{S}_1 \cdot \mathbf{B}_{1,\mathsf{st}_1} + \mathbf{E}_{1,\mathsf{st}_1}$ where $\mathbf{E}_{1,\mathsf{st}_1}$ is a submatrix of $\mathbf{E}_1$. Thus, we could write the following

$$\|\mathbf{Err}_1\|_\infty = \left\|\mathbf{\Delta}_1 - \widetilde{\mathbf{\Delta}}_1\right\|_\infty = \|\mathbf{E}_{1,\mathsf{st}_1}\|_\infty \leq m \cdot \sigma < m^2 \cdot \sigma.$$

This completes the proof of base case. For the induction step, we assume that the above lemma holds for $j^* - 1$, and show that it holds for $j^*$ as well.

**Induction Step.** We know that $\mathbf{\Delta}_{j^*} = \mathbf{\Delta}_{j^*-1} \cdot \mathbf{C}_{j^*}$. Also, $\mathbf{\Delta}_{j^*-1} = \widetilde{\mathbf{\Delta}}_{j^*-1} + \mathbf{Err}_{j^*-1}$. So, we could write the following

$$\begin{aligned}
\mathbf{\Delta}_{j^*} &= \widetilde{\mathbf{\Delta}}_{j^*-1} \cdot \mathbf{C}_{j^*} + \mathbf{Err}_{j^*-1} \cdot \mathbf{C}_{j^*} \\
&= \mathbf{\Gamma}_{j^*-1} \cdot \left(\mathbf{B}_{j^*-1,\mathsf{st}_{j^*-1}} \cdot \mathbf{C}_{j^*}\right) + \mathbf{Err}_{j^*-1} \cdot \mathbf{C}_{j^*} \\
&= \mathbf{\Gamma}_{j^*-1} \cdot \left(\mathbf{S}_{j^*} \cdot \mathbf{B}_{j^*,\mathsf{st}_{j^*}} + \mathbf{E}_{j^*,\mathsf{st}_{j^*-1}}\right) + \mathbf{Err}_{j^*-1} \cdot \mathbf{C}_{j^*} \\
&= \widetilde{\mathbf{\Delta}}_{j^*} + \mathbf{\Gamma}_{j^*-1} \cdot \mathbf{E}_{j^*,\mathsf{st}_{j^*-1}} + \mathbf{Err}_{j^*-1} \cdot \mathbf{C}_{j^*}
\end{aligned}$$

Here, $\mathbf{E}_{j^*,\mathsf{st}_{j^*-1}}$ is an $n \times m$ submatrix of $\mathbf{E}_{j^*}$. Finally, we can bound $\mathbf{Err}_{j^*}$ as follows

$$\begin{aligned}
\|\mathbf{Err}_{j^*}\|_\infty &= \left\|\mathbf{\Delta}_{j^*} - \widetilde{\mathbf{\Delta}}_{j^*}\right\|_\infty = \left\|\mathbf{\Gamma}_{j^*-1} \cdot \mathbf{E}_{j^*,\mathsf{st}_{j^*-1}} + \mathbf{Err}_{j^*-1} \cdot \mathbf{C}_{j^*}\right\|_\infty \\
&\leq \left\|\mathbf{\Gamma}_{j^*-1} \cdot \mathbf{E}_{j^*,\mathsf{st}_{j^*-1}}\right\|_\infty + \|\mathbf{Err}_{j^*-1} \cdot \mathbf{C}_{j^*}\|_\infty \\
&\leq \left(n^2 \cdot \sigma\right)^{j^*-1} \cdot m \cdot \sigma + (j^*-1) \cdot (m^2 \cdot \sigma)^{j^*-1} \cdot m^2 \cdot \sigma \leq j^* \cdot (m^2 \cdot \sigma)^{j^*}
\end{aligned}$$

This completes the proof of above claim. ■

The remaining proof of the lemma will have two parts, the first part dealing with the case when $\mathsf{BP}^{(i)}(z) = \beta_i$ for all $i$, and the second one dealing with when $\mathsf{BP}^{(i)}(z) \neq \beta_i$ for some $i$.

Recall that the $\mathsf{Comp\text{-}Eval}$ algorithm computes matrix $\mathbf{M} = \sum_{i=1}^{\ell_{\mathrm{PRG}}} \mathbf{\Delta}_L^{(i)}$. Let $\widetilde{\mathbf{M}} = \sum_{i=1}^{\ell_{\mathrm{PRG}}} \widetilde{\mathbf{\Delta}}_L^{(i)}$ and $\mathsf{Err} = \sum_{i=1}^{\ell_{\mathrm{PRG}}} \mathbf{Err}_L^{(i)}$. Also, we parse these matrices as $\mathbf{M} = \left[\mathbf{M}^{(1)} \,||\, \mathbf{M}^{(2)}\right]$, $\widetilde{\mathbf{M}} = \left[\widetilde{\mathbf{M}}^{(1)} \,||\, \widetilde{\mathbf{M}}^{(2)}\right]$ and $\mathsf{Err} = \left[\mathsf{Err}^{(1)} \,||\, \mathsf{Err}^{(2)}\right]$, where $\mathbf{M}^{(1)}, \widetilde{\mathbf{M}}^{(1)}$ and $\mathsf{Err}^{(1)}$ are $n \times n$ (square) matrices.

First, note that $\mathbf{M} = \widetilde{\mathbf{M}} + \mathsf{Err}$. Also, using Claim 4.1, we can write that

$$\|\mathsf{Err}\|_\infty = \left\|\sum_{i=1}^{\ell_{\mathrm{PRG}}} \left(\mathbf{\Delta}_L^{(i)} - \widetilde{\mathbf{\Delta}}_L^{(i)}\right)\right\|_\infty \leq \sum_{i=1}^{\ell_{\mathrm{PRG}}} \left\|\mathbf{\Delta}_L^{(i)} - \widetilde{\mathbf{\Delta}}_L^{(i)}\right\|_\infty \leq \ell_{\mathrm{PRG}} \cdot L \cdot \left(m^2 \cdot \sigma\right)^L = \mathsf{Bd}.$$

Next, consider the following scenarios.

**Part 1: $\mathsf{BP}^{(i)}(z) = \beta_i$ for all $i \leq \ell_{\mathrm{PRG}}$.** First, recall that the top level matrices always satisfy the following constraints during honest obfuscation:

$$\sum_{i\,:\,\beta_i=0} \mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}} + \sum_{i\,:\,\beta_i=1} \mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}} = \begin{cases} \mathbf{0}^{n\times m} & \text{if } \mathsf{msg}=0 \\ \sqrt{q} \cdot \left[\mathbf{I}_n \,||\, \mathbf{0}^{n\times(m-n)}\right] & \text{if } \mathsf{msg}=1. \end{cases}$$

Since $\mathsf{BP}^{(i)}(z) = \beta_i$ for all $i \leq \ell_{\mathrm{PRG}}$, we can alternatively write it as

$$\sum_{i=1}^{\ell_{\mathrm{PRG}}} \mathbf{B}^{(i)}_{L,\mathsf{st}_L^{(i)}} = \begin{cases} \mathbf{0}^{n\times m} & \text{if } \mathsf{msg}=0 \\ \sqrt{q} \cdot \left[\mathbf{I}_n \,||\, \mathbf{0}^{n\times(m-n)}\right] & \text{if } \mathsf{msg}=1. \end{cases}$$

Note that

$$\widetilde{\mathbf{M}} = \sum_{i=1}^{\ell_{\mathrm{PRG}}} \widetilde{\mathbf{\Delta}}_L^{(i)} = \sum_{i=1}^{\ell_{\mathrm{PRG}}} \mathbf{\Gamma}_L \cdot \mathbf{B}^{(i)}_{L,\mathsf{st}_L^{(i)}} = \mathbf{\Gamma}_L \cdot \sum_{i=1}^{\ell_{\mathrm{PRG}}} \mathbf{B}^{(i)}_{L,\mathsf{st}_L^{(i)}} = \begin{cases} \mathbf{0}^{n\times m} & \text{if } \mathsf{msg}=0 \\ \sqrt{q} \cdot \left[\mathbf{\Gamma}_L \,||\, \mathbf{0}^{n\times(m-n)}\right] & \text{if } \mathsf{msg}=1. \end{cases}$$

Next, we consider the following two cases dependending upon the message being obfuscated — (1) $\mathsf{msg}=0$, (2) $\mathsf{msg}=1$.

**Case 1** ($\mathsf{msg} = 0$)**.** We already know that $\|\mathsf{Err}\|_\infty \leq \mathsf{Bd}$. Now since in this case $\widetilde{\mathbf{M}} = \mathbf{0}^{n\times m}$, we can say that

$$\|\mathbf{M}\|_\infty = \left\|\widetilde{\mathbf{M}} + \mathsf{Err}\right\|_\infty = \|\mathsf{Err}\|_\infty \leq \mathsf{Bd}.$$

Thus, matrix $\mathbf{M}$ (computed during evaluation) always satisfies the condition that $\|\mathbf{M}\|_\infty \leq \mathsf{Bd}$ if $\mathsf{msg} = 0$.

**Case 2** ($\mathsf{msg} = 1$)**.** First, note that during obfuscation we sample secret matrices $\mathbf{S}^{(b)}_{\mathsf{level}}$ (for each $\mathsf{level}$ and bit $b$) such that they are short and *always* invertible. Therefore, matrix $\mathbf{\Gamma}_L$ (which is product of $L$ seret matrices) is also invertible. Thus, we can write that

$$\|\mathbf{\Gamma}_L\|_\infty \geq 1, \quad \|\mathbf{\Gamma}_L\|_\infty = \left\|\prod_{j=1}^{L} \mathbf{S}_j\right\|_\infty \leq n^L \prod_{j=1}^{L} \|\mathbf{S}_j\|_\infty \leq (m^2 \cdot \sigma)^L < \mathsf{Bd}.$$

The lower bound of 1 follows from the fact that $\mathbf{\Gamma}_L$ is non-singular (and integral) matrix, and upper bound follows from the fact that matrices $\mathbf{S}^{(b)}_{\mathsf{level}}$ are sampled from bounded Gaussian distribution.

Since $\mathsf{msg} = 1$, we know that $\widetilde{\mathbf{M}}^{(1)} = \sqrt{q} \cdot \mathbf{\Gamma}_L$ and $\widetilde{\mathbf{M}}^{(2)} = \mathbf{0}^{n\times(m-n)}$. Using all these facts, we can show that if $\mathsf{msg} = 1$, then the checks during component evaluation always succeed. More formally, first we show that $\mathbf{M}^{(2)}$ is a short matrix.

$$\left\|\mathbf{M}^{(2)}\right\|_\infty = \left\|\widetilde{\mathbf{M}}^{(2)} + \mathsf{Err}^{(2)}\right\|_\infty = \left\|\mathsf{Err}^{(2)}\right\|_\infty \leq \|\mathsf{Err}\|_\infty \leq \mathsf{Bd}.$$

Next, we show that matrix $\mathbf{M}^{(1)}$ has large entries. In other words, matrix $\mathbf{M}$ has high $l_\infty$ norm. Concretely,

$$\|\mathbf{M}\|_\infty = \left\|\widetilde{\mathbf{M}} + \mathsf{Err}\right\|_\infty \leq \left\|\widetilde{\mathbf{M}}\right\|_\infty + \|\mathsf{Err}\|_\infty = \left\|\widetilde{\mathbf{M}}^{(1)}\right\|_\infty + \|\mathsf{Err}\|_\infty = \|\sqrt{q} \cdot \mathbf{\Gamma}_L\|_\infty + \|\mathsf{Err}\|_\infty \leq \sqrt{q} \cdot \mathsf{Bd} + \mathsf{Bd}.$$

$$\|\mathbf{M}\|_\infty = \left\|\widetilde{\mathbf{M}} + \mathsf{Err}\right\|_\infty \geq \left\|\widetilde{\mathbf{M}}\right\|_\infty - \|\mathsf{Err}\|_\infty \geq \left\|\widetilde{\mathbf{M}}^{(1)}\right\|_\infty - \|\mathsf{Err}\|_\infty = \|\sqrt{q} \cdot \mathbf{\Gamma}_L\|_\infty - \|\mathsf{Err}\|_\infty \geq \sqrt{q} - \mathsf{Bd}.$$

Therefore, $\|\mathbf{M}\|_\infty \in \left[\sqrt{q} - \mathsf{Bd}, (\sqrt{q}+1) \cdot \mathsf{Bd}\right]$. Thus, if $\mathsf{msg} = 1$ then the evaluation always outputs 1.

**Part 2:** $\mathsf{BP}^{(i)}(z) \neq \beta_i$ **for some** $i \leq \ell_{\mathrm{PRG}}$**.** Let $k$ be the first such index. Then $\mathbf{B}^{(k)}_{L,\mathsf{st}^{(k)}_L}$ is a uniformly random $n \times m$ matrix, independent of the other top level matrices. As a result, $\sum_{i=1}^{\ell_{\mathrm{PRG}}} \mathbf{B}^{(i)}_{L,\mathsf{st}^{(i)}_L}$ is a uniformly random matrix. As we noted before $\mathbf{\Gamma}_L$ is an invertible matrix, therefore $\widetilde{\mathbf{M}} = \sum_{i=1}^{\ell_{\mathrm{PRG}}} \widetilde{\mathbf{\Delta}}^{(i)}_L = \mathbf{\Gamma}_L \cdot \sum_{i=1}^{\ell_{\mathrm{PRG}}} \mathbf{B}^{(i)}_{L,\mathsf{st}^{(i)}_L}$ is a uniformly random matrix. This also implies that sub-matrices $\widetilde{\mathbf{M}}^{(1)}$ and $\widetilde{\mathbf{M}}^{(2)}$ are random matrices as well.

Therefore, with all but negligible probability, both $\|\mathbf{M}\|_\infty$ as well as $\left\|\mathbf{M}^{(2)}\right\|_\infty$ will not be small (i.e., less than $\mathsf{Bd}$). Thus, in this scenario, the evaluator will output $\perp$ (with all but negligible probability).

This concludes the proof of Lemma 4.1.

■

Using the above lemma, we can now argue the correctness of our scheme. First, we need to show correctness for the case when $P(x) = \alpha$.

**Claim 4.2.** For all security parameters $\lambda$, inputs $x \in \{0,1\}^{\ell_{\mathrm{in}}}$, programs $P \in \mathcal{C}_{\ell_{\mathrm{in}},\ell_{\mathrm{out}},d}$ and messages $\mathsf{msg} \in \{0,1\}$, if $P(x) = \alpha$, then

$$\mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha)), x) = \mathsf{msg}.$$

*Proof.* First, the obfuscator encrypts the program $P$ using an LHE secret key $\mathsf{lhe.sk}$, and sets $\mathsf{ct} \leftarrow \mathsf{LHE.Enc}(\mathsf{lhe.sk}, P)$. The evaluator evaluates the LHE ciphertext on universal circuit $U_x(\cdot)$, which results in an evaluated ciphertext $\widetilde{\mathbf{ct}}$. Now, by the correctness of the LHE scheme, decryption of $\widetilde{\mathbf{ct}}$ using $\mathsf{lhe.sk}$ outputs $\alpha$. Therefore, $\mathrm{PRG}(h(\mathsf{LHE.Dec}(\mathsf{lhe.sk}, \widetilde{\mathbf{ct}}))) = \beta$.[11] Then, using Lemma 4.1, we can argue that $\mathsf{Comp\text{-}Eval}$ outputs $\mathsf{msg}$, and thus $\mathsf{Eval}$ outputs $\mathsf{msg}$. ∎

**Claim 4.3.** For all security parameters $\lambda$, inputs $x \in \{0,1\}^{\ell_{\text{in}}}$, programs $P \in \mathcal{C}_{\ell_{\text{in}},\ell_{\text{out}},d}$, $\alpha \in \{0,1\}^{\ell_{\text{out}}}$ such that $P(x) \neq \alpha$ and $\mathsf{msg} \in \{0,1\}$,

$$\Pr[\mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha), x) = \perp] \geq 1 - \mathrm{negl}(\lambda)$$

where the probability is taken over the random coins used during obfuscation.

*Proof.* Fix any security parameter $\lambda$, program $P$, $\alpha$, $x$ such that $P(x) \neq \alpha$ and message $\mathsf{msg}$. Let us now consider the events $\mathsf{Error} : \mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha), x) \neq \perp$ and $\mathsf{Bad} : \mathrm{PRG}(h(P(x))) = \mathrm{PRG}(h(\alpha))$. Clearly, $\Pr[\mathsf{Error}] \leq \Pr[\mathsf{Error} \mid \overline{\mathsf{Bad}}] + \Pr[\mathsf{Bad}]$. We will prove that both terms $\Pr[\mathsf{Bad}]$ and $\Pr[\mathsf{Error} \mid \overline{\mathsf{Bad}}]$ are both negligible in the security parameter.

$\Pr[\mathsf{Error} \mid \overline{\mathsf{Bad}}]$ : Since we are conditioning on $\overline{\mathsf{Bad}}$, it follows that $\mathrm{PRG}(h(P(x))) \neq \mathrm{PRG}(h(\alpha))$, and note that this probability is only over the coins chosen during $\mathsf{Comp\text{-}Gen}$ and $\mathsf{Comp\text{-}Eval}$. Let $\mathsf{ct}$ be the LHE encryption of $P$, and $\widetilde{\mathbf{ct}} = \mathsf{LHE.Eval}(\mathsf{ek}, U_x(\cdot), \mathsf{ct})$. As described in the construction, let $Q$ be the program that, on input a LHE ciphertext, first decrypts it using $\mathsf{sk}$, then computes $\mathrm{PRG}(h(y))$ (where $y$ is the decrypted string). Let $\mathsf{BP}^{(i)}$ denote the branching program computing the $i^{th}$ output bit of $Q$. Using the correctness of LHE decryption, it follows that $Q(\widetilde{\mathbf{ct}}) = \mathrm{PRG}(h(P(x))) \neq \mathrm{PRG}(h(\alpha))$, and therefore there exists some $i \leq \ell_{\text{PRG}}$ such that $\mathsf{BP}^{(i)}(\widetilde{\mathbf{ct}}) \neq \beta_i$ (recall $\beta = \mathrm{PRG}(h(\alpha))$). As a result, using Lemma 4.1, we can conclude that for all $\mathsf{msg} \in \{0,1\}$, $\Pr[\mathsf{Comp\text{-}Eval}(\widetilde{\mathbf{ct}}, \mathsf{Comp\text{-}Gen}(\{\mathsf{BP}^{(i)}\}_i, \beta, \mathsf{msg})) \neq \perp] \leq \mathrm{negl}(\lambda)$.

$\Pr[\mathsf{Bad}]$ : For any $y \in \{0,1\}^{\ell_{\text{PRG}}}$, let $\mathrm{PRG}^{-1}(y) = \{z \in \{0,1\}^{\ell_{\mathcal{H}}} \text{ s.t. } \mathrm{PRG}(z) = y\}$. Using the PRG security, we can argue that, for any $y \in \{0,1\}^{\ell_{\text{PRG}}}$, $|\mathrm{PRG}^{-1}(y)|/2^{\ell_{\mathcal{H}}} \leq \mathrm{negl}(\lambda)$. Suppose, on the contrary, there exists some $y^*$ such that $\mathrm{PRG}^{-1}(y^*)/2^{\ell_{\mathcal{H}}} = \epsilon$, where $\epsilon$ is non-negligible, then we can construct a PPT adversary that can break the PRG security with advantage $\epsilon^2 - \mathrm{negl}(\lambda)$. Concretely, the adversary receives a challenge string $t$. It chooses a uniformly random string $s \leftarrow \{0,1\}^{\ell_{\mathcal{H}}}$ and checks if $\mathrm{PRG}(s) = t$. If so, it outputs that $t$ is a pseudorandom string, else it guesses randomly. Clearly, the advantage of this attacker is at least $\epsilon^2 - 2^{-(\ell_{\text{PRG}} - \ell_{\mathcal{H}})}$, where $2^{-(\ell_{\text{PRG}} - \ell_{\mathcal{H}})}$ loss is in the case when the challenge string $t$ lies in the image space of PRG. Since, $\ell_{\text{PRG}} - \ell_{\mathcal{H}} = \omega(\log \lambda)$. Therefore, if PRG is secure then for every $y \in \{0,1\}^{\ell_{\text{PRG}}}$, $\dfrac{|\mathrm{PRG}^{-1}(y)|}{2^{\ell_{\mathcal{H}}}} \leq \mathrm{negl}(\lambda)$.

Let $\phi_\gamma = \mathrm{PRG}^{-1}(\mathrm{PRG}(\gamma))$. Now we consider the probability $\Pr[\mathsf{Bad}]$, where the probability is over the choice of $h \leftarrow \mathcal{H}$.

$$\begin{aligned}
\Pr[\mathsf{Bad}] &= \Pr[\mathrm{PRG}(h(P(x))) = \mathrm{PRG}(h(\alpha))] \\
&= \sum_{\gamma \in \{0,1\}^{\ell_{\mathcal{H}}}} \Pr\left[h(\alpha) = \gamma \wedge h(P(x)) \in \phi_\gamma\right] \\
&= \sum_{\gamma \in \{0,1\}^{\ell_{\mathcal{H}}}} \sum_{\delta \in \phi_\gamma} \Pr\left[h(\alpha) = \gamma \wedge h(P(x)) = \delta\right] \\
&\leq \sum_{\gamma \in \{0,1\}^{\ell_{\mathcal{H}}}} \sum_{\delta \in \phi_\gamma} \frac{1}{\left(2^{\ell_{\mathcal{H}}}\right)^2} = \sum_{\gamma \in \{0,1\}^{\ell_{\mathcal{H}}}} \frac{|\phi_\gamma|}{\left(2^{\ell_{\mathcal{H}}}\right)^2} \\
&\leq \max_\gamma \frac{|\phi_\gamma|}{2^{\ell_{\mathcal{H}}}} \leq \mathrm{negl}(\lambda).
\end{aligned}$$

[11] As before, we are overloading the notation and using $\mathsf{LHE.Dec}$ to decrypt multiple ciphertexts.

Therefore, $\Pr[\mathsf{Error}]$ is bounded by a negligible function in $\lambda$. This concludes the proof. ■

## 4.2 Security

We will now prove the scheme secure as per Definition 3.4. For proving security, we will first need to define a PPT simulator. The simulator $\mathsf{Sim}$ gets as input the security parameter $\lambda$ and size of program $1^{|P|}$, and it must output a simulated program.

We emphasize that our proof will use a polynomial number of hybrids and each (computational) reduction will only rely on a primitive being secure against poly-time attackers. Thus we avoid making any subexponential hardness assumptions.

### 4.2.1 Simulator $\mathsf{Sim}$

The simulator first chooses the parameters $n, m, q, \sigma, \ell_{\mathrm{PRG}}$ as in the original scheme. Next, it chooses LHE secret/evaluation keys. It computes $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^{d \log d})$ (note that the depth $d$ of the circuit class is fixed for the scheme). It then computes an encryption of $\mathbf{0}^{|P|}$. Let $\mathsf{ct} \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}, \mathbf{0}^{|P|})$. Finally, the simulator chooses $\ell_{\mathrm{PRG}}$ matrices $\mathbf{B}_{0,1}^{(i)} \leftarrow \mathbb{Z}_q^{n \times m}$ and low norm matrices $\{(\mathbf{C}_j^{(i,0)}, \mathbf{C}_j^{(i,1)})\}_{i,j}$ for $i \leq \ell_{\mathrm{PRG}}$, $j \leq L$ where $\mathbf{C}_j^{(i,b)} \leftarrow \chi^{m \times m}$. The obfuscation consists of the LHE evaluation key $\mathsf{ek}$, ciphertext $\mathsf{ct}$, together with the components $\left( \left\{ \mathbf{B}_{0,1}^{(i)} \right\}_i, \left\{ (\mathbf{C}_j^{(i,0)}, \mathbf{C}_j^{(i,1)}) \right\}_{i,j} \right)$.

To prove security, we will define a sequence of hybrids, and show that the hybrids are computationally indistinguishable. The first hybrid corresponds to the case when the challenger outputs an obfuscation of the program, while the final hybrid corresponds to one where the challenger outputs an obfuscation of an 'all-reject' program. At a high level, our aim is to replace the LHE encryption of the program $P$ with an encryption of all-zeroes string, and to remove the message $\mathsf{msg}$ from the matrix components. To achieve this, we will first replace the matrix components with random low-norm matrices. This switch ensures that the matrix components contain no information about the $\mathsf{msg}$ or the LHE secret key. We can then switch the encryption of $P$ with encryption of all-zeroes string.

### 4.2.2 Sequence of Hybrid Games

In the security game, the adversary $\mathcal{A}$ starts by choosing a program $P \in \mathcal{C}_{\ell_{\mathrm{in}}, \ell_{\mathrm{out}}, d}$ and message $\mathsf{msg} \in \{0,1\}$. It sends the $P, \mathsf{msg}$ to the challenger and the challenger chooses a random bit $\zeta$. If $\zeta = 0$, the challenger computes program $\widetilde{P}$ as an obfuscation of $P$ for message $\mathsf{msg}$ with random lock using the obfuscation algorithm. Otherwise, the challenger computes $\widetilde{P}$ using the simulator. Now, in the sequence of hybrids described below, $\mathsf{Game}$ 0 will correspond to the case $\zeta = 0$ (i.e., the challenger computes the obfuscated program honestly) and (final) $\mathsf{Game}$ 5 will correspond to $\zeta = 1$ (i.e., the challenger simulates the obfuscated program).

In the following hybrid description, we only describe the challenger's execution and in the later games, we only highlight the changes made in each consecutive hybrid. Note that the program $P$ and message $\mathsf{msg}$ are provided by $\mathcal{A}$ to the challenger.

$\mathsf{Game}$ 0**:** As mentioned above, in this game the challenger computes the obfuscated program honestly.

1. The challenger first chooses the LWE parameters $n$, $m$, $q$, $\sigma$, $\chi$ and $\ell_{\mathrm{PRG}}$. Recall $L$ denotes the length of the branching programs.
2. It chooses $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^{d \log d})$ and sets $\mathsf{ct} \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}, P)$.
3. Next, it chooses a uniformly random string $\alpha \leftarrow \{0,1\}^{\ell_{\mathrm{out}}}$, hash function $h \leftarrow \mathcal{H}$ and sets $\beta = \mathrm{PRG}(h(\alpha))$.

4. Next, consider the following program $Q$. It takes as input an LHE ciphertext $\mathsf{ct}$, has $\mathsf{sk}$ hardwired and does the following: it decrypts the input ciphertext $\mathsf{ct}$ to get string $x$ and outputs $\mathrm{PRG}(h(x))$. For $i \leq \ell_{\mathrm{PRG}}(\lambda)$, let $\mathsf{BP}^{(i)}$ denote the branching program that outputs the $i^{th}$ bit of $\mathrm{PRG}(h(x))$.
5. It chooses $\ell_{\mathrm{PRG}}$ uniformly random matrices $\mathbf{B}_L^{(i)}$ of dimensions $5n \times m$, such that the following constraint is satisfied (recall $\mathbf{B}_{L,1}^{(i)}$ represents the first $n$ rows of $\mathbf{B}_L^{(i)}$, $\mathbf{B}_{L,2}^{(i)}$ represents the next $n$ rows of $\mathbf{B}_L^{(i)}$, etc)

$$\sum_{i \,:\, \beta_i = 0} \mathbf{B}_{L,\mathsf{rej}^{(i)}}^{(i)} + \sum_{i \,:\, \beta_i = 1} \mathbf{B}_{L,\mathsf{acc}^{(i)}}^{(i)} = \begin{cases} \mathbf{0} & \text{if } \mathsf{msg} = 0. \\ \sqrt{q} \cdot \left[\mathbf{I}_n \,||\, \mathbf{0}^{n \times (m-n)}\right] & \text{if } \mathsf{msg} = 1. \end{cases}$$

6. For $i = 1$ to $\ell_{\mathrm{PRG}}$ and $j = 0$ to $L-1$, it chooses $(\mathbf{B}_j^{(i)}, T_j^{(i)}) \leftarrow \mathsf{TrapGen}(1^{5n}, 1^m, q)$.
7. Next, it generates the components for each level. For each $i \in [1, \ell_{\mathrm{PRG}}]$ and each level $\mathsf{level} \in [1, L]$, do the following:
   (a) Choose matrices $\mathbf{S}_{\mathsf{level}}^{(0)}, \mathbf{S}_{\mathsf{level}}^{(1)} \leftarrow \chi^{n \times n}$ and $\mathbf{E}_{\mathsf{level}}^{(i,0)}, \mathbf{E}_{\mathsf{level}}^{(i,1)} \leftarrow \chi^{5n \times m}$ for $i \leq \ell_{\mathrm{PRG}}$. If either $\mathbf{S}_{\mathsf{level}}^{(0)}$ or $\mathbf{S}_{\mathsf{level}}^{(1)}$ has determinant zero, then set it to be $\mathbf{I}_n$.
   (b) For $b \in \{0,1\}$, set matrix $\mathbf{D}_{\mathsf{level}}^{(i,b)}$ as a permutation of the matrix blocks of $\mathbf{B}_{\mathsf{level}}^{(i)}$ according to the permutation $\sigma_{\mathsf{level},b}^{(i)}(\cdot)$.
   (c) Set $\mathbf{M}_{\mathsf{level}}^{(i,b)} = \left(\mathbf{I}_5 \otimes \mathbf{S}_{\mathsf{level}}^{(b)}\right) \cdot \mathbf{D}_{\mathsf{level}}^{(i,b)} + \mathbf{E}_{\mathsf{level}}^{(i,b)}$ for $i \leq \ell_{\mathrm{PRG}}$.
   (d) Compute $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{\mathsf{level}-1}^{(i)}, T_{\mathsf{level}-1}^{(i)}, \sigma, \mathbf{M}_{\mathsf{level}}^{(i,b)})$
8. The final obfuscated program consists of the LHE evaluation key $\mathsf{ek}$, LHE encryption $\mathsf{ct}$, together with the components $\left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, \left\{(\mathbf{C}_j^{(i,0)}, \mathbf{C}_j^{(i,1)})\right\}_{i,j}\right)$.

**Game 1:** In this experiment, the challenger replaces $h(\alpha)$ with a uniformly random string $\alpha'$. Also, it chooses the matrices $\mathbf{S}_{\mathsf{level}}^{(0)}, \mathbf{S}_{\mathsf{level}}^{(1)}$ without checking if their determinant is non-zero.

3. Next, it chooses a hash function $h \leftarrow \mathcal{H}$, a uniformly random string $\alpha' \leftarrow \{0,1\}^{\ell_{\mathcal{H}}}$ and sets $\beta = \mathrm{PRG}(\alpha')$.

7. Next, it generates the components for each level. For each level $\mathsf{level} \in [1, L]$, do the following:
   (a) Choose matrices $\mathbf{S}_{\mathsf{level}}^{(0)}, \mathbf{S}_{\mathsf{level}}^{(1)} \leftarrow \chi^{n \times n}$ and $\mathbf{E}_{\mathsf{level}}^{(i,0)}, \mathbf{E}_{\mathsf{level}}^{(i,1)} \leftarrow \chi^{5n \times m}$ for $i \leq \ell_{\mathrm{PRG}}$.
   (b) For $b \in \{0,1\}$, set matrix $\mathbf{D}_{\mathsf{level}}^{(i,b)}$ as a permutation of the matrix blocks of $\mathbf{B}_{\mathsf{level}}^{(i)}$ according to the permutation $\sigma_{\mathsf{level},b}^{(i)}(\cdot)$.
   (c) Set $\mathbf{M}_{\mathsf{level}}^{(i,b)} = \left(\mathbf{I}_5 \otimes \mathbf{S}_{\mathsf{level}}^{(b)}\right) \cdot \mathbf{D}_{\mathsf{level}}^{(i,b)} + \mathbf{E}_{\mathsf{level}}^{(i,b)}$ for $i \leq \ell_{\mathrm{PRG}}$.
   (d) Compute $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{\mathsf{level}-1}^{(i)}, T_{\mathsf{level}-1}^{(i)}, \sigma, \mathbf{M}_{\mathsf{level}}^{(i,b)})$

**Game 2:** In this experiment, the challenger replaces the pseudorandom string $\beta = \mathrm{PRG}(\alpha')$ with a truly random string.

3. Next, it chooses a hash function $h \leftarrow \mathcal{H}$, a uniformly random string $\beta \leftarrow \{0,1\}^{\ell_{\mathrm{PRG}}}$.

**Game 3:** In this experiment, the challenger chooses the top level matrices uniformly at random.

5. It chooses $\ell_{\mathrm{PRG}}$ uniformly random matrices $\mathbf{B}_L^{(i)}$ of dimensions $5n \times m$, without any constraints.

We will now define $4L+1$ intermediate game $\mathsf{Game}\ (4, j^*, 0), \mathsf{Game}\ (4, j^*, 1), \mathsf{Game}\ (4, j^*, 2), \mathsf{Game}\ (4, j^*, 3)$ for $j^* \in \{0, 1, \ldots, L-1\}$ and $\mathsf{Game}\ (4, L, 0)$. The experiment $\mathsf{Game}\ (4, 0, 0)$ will be same as $\mathsf{Game}\ 3$.

**Game $(4, j^*, 0)$:** In this experiment, the challenger chooses the top $j^*+1$ matrices $\mathbf{B}_j^{(i)}$ uniformly at random (without a trapdoor). Also, the top $j^*$ $\mathbf{C}_j^{(i,b)}$ matrices, for $j > L - j^*$, are chosen randomly from the noise distribution $\chi^{m \times m}$.

6. For $i = 1$ to $\ell_{\mathrm{PRG}}$ and $j = 0$ to $L-1$,
   - if $j < L - j^*$ it chooses $(\mathbf{B}_j^{(i)}, T_j^{(i)}) \leftarrow \mathsf{TrapGen}(1^{5n}, 1^m, q)$.
   - if $j \geq L - j^*$, it chooses $\mathbf{B}_j^{(i)}$ uniformly at random from $\mathbb{Z}_q^{5n \times m}$.
7. Next, it generates the components for each level. For each level $\mathsf{level} \in [1, L-j^*]$, do the following:
   (a) Choose matrices $\mathbf{S}_{\mathsf{level}}^{(0)}, \mathbf{S}_{\mathsf{level}}^{(1)} \leftarrow \chi^{n \times n}$ and $\mathbf{E}_{\mathsf{level}}^{(i,0)}, \mathbf{E}_{\mathsf{level}}^{(i,1)} \leftarrow \chi^{5n \times m}$ for $i \leq \ell_{\mathrm{PRG}}$.
   (b) For $b \in \{0,1\}$, set matrix $\mathbf{D}_{\mathsf{level}}^{(i,b)}$ as a permutation of the matrix blocks of $\mathbf{B}_{\mathsf{level}}^{(i)}$ according to the permutation $\sigma_{\mathsf{level},b}^{(i)}(\cdot)$.
   (c) Set $\mathbf{M}_{\mathsf{level}}^{(i,b)} = \left(\mathbf{I}_5 \otimes \mathbf{S}_{\mathsf{level}}^{(b)}\right) \cdot \mathbf{D}_{\mathsf{level}}^{(i,b)} + \mathbf{E}_{\mathsf{level}}^{(i,b)}$ for $i \leq \ell_{\mathrm{PRG}}$.
   (d) Compute $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{\mathsf{level}-1}^{(i)}, T_{\mathsf{level}-1}^{(i)}, \sigma, \mathbf{M}_{\mathsf{level}}^{(i,b)})$

   For all $\mathsf{level} > L - j^*$, $i \leq \ell_{\mathrm{PRG}}$ and $b \in \{0,1\}$ choose $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \chi^{m \times m}$.

**Game $(4, j^*, 1)$:** In this experiment, the challenger chooses $\mathbf{M}_{\mathsf{level}}^{(i,0)}$ uniformly at random for $\mathsf{level} = L - j^*$ ($\mathbf{M}_{\mathsf{level}}^{(i,1)}$ is same as before).

7. Next, it generates the components for each level. For each level $\mathsf{level} \in [1, L-j^*]$, do the following:
   (a) Choose matrices $\mathbf{S}_{\mathsf{level}}^{(0)}, \mathbf{S}_{\mathsf{level}}^{(1)} \leftarrow \chi^{n \times n}$ and $\mathbf{E}_{\mathsf{level}}^{(i,0)}, \mathbf{E}_{\mathsf{level}}^{(i,1)} \leftarrow \chi^{5n \times m}$ for $i \leq \ell_{\mathrm{PRG}}$.
   (b) For $b \in \{0,1\}$, set matrix $\mathbf{D}_{\mathsf{level}}^{(i,b)}$ as a permutation of the matrix blocks of $\mathbf{B}_{\mathsf{level}}^{(i)}$ according to the permutation $\sigma_{\mathsf{level},b}^{(i)}(\cdot)$.
   (c) If $\mathsf{level} = L - j^*$, choose $\mathbf{M}_{\mathsf{level}}^{(i,0)} \leftarrow \mathbb{Z}_q^{5n \times m}$ and $\mathbf{M}_{\mathsf{level}}^{(i,1)}$ same as previous hybrid.
   Else set $\mathbf{M}_{\mathsf{level}}^{(i,b)} = \left(\mathbf{I}_5 \otimes \mathbf{S}_{\mathsf{level}}^{(b)}\right) \cdot \mathbf{D}_{\mathsf{level}}^{(i,b)} + \mathbf{E}_{\mathsf{level}}^{(i,b)}$ for $i \leq \ell_{\mathrm{PRG}}$.
   (d) Compute $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{\mathsf{level}-1}^{(i)}, T_{\mathsf{level}-1}^{(i)}, \sigma, \mathbf{M}_{\mathsf{level}}^{(i,b)})$

   For all $\mathsf{level} > L - j^*$, $i \leq \ell_{\mathrm{PRG}}$ and $b \in \{0,1\}$ choose $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \chi^{m \times m}$.

**Game $(4, j^*, 2)$:** In this experiment, the challenger chooses $\mathbf{M}_{\mathsf{level}}^{(i,b)}$ uniformly at random for $\mathsf{level} = L - j^*$ and $b \in \{0,1\}$.

7. Next, it generates the components for each level. For each level $\mathsf{level} \in [1, L-j^*]$, do the following:
   (a) Choose matrices $\mathbf{S}_{\mathsf{level}}^{(0)}, \mathbf{S}_{\mathsf{level}}^{(1)} \leftarrow \chi^{n \times n}$ and $\mathbf{E}_{\mathsf{level}}^{(i,0)}, \mathbf{E}_{\mathsf{level}}^{(i,1)} \leftarrow \chi^{5n \times m}$ for $i \leq \ell_{\mathrm{PRG}}$.
   (b) For $b \in \{0,1\}$, set matrix $\mathbf{D}_{\mathsf{level}}^{(i,b)}$ as a permutation of the matrix blocks of $\mathbf{B}_{\mathsf{level}}^{(i)}$ according to the permutation $\sigma_{\mathsf{level},b}^{(i)}(\cdot)$.
   (c) If $\mathsf{level} = L - j^*$, choose $\mathbf{M}_{\mathsf{level}}^{(i,b)} \leftarrow \mathbb{Z}_q^{5n \times m}$.
   Else set $\mathbf{M}_{\mathsf{level}}^{(i,b)} = \left(\mathbf{I}_5 \otimes \mathbf{S}_{\mathsf{level}}^{(b)}\right) \cdot \mathbf{D}_{\mathsf{level}}^{(i,b)} + \mathbf{E}_{\mathsf{level}}^{(i,b)}$ for $i \leq \ell_{\mathrm{PRG}}$.
   (d) Compute $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{\mathsf{level}-1}^{(i)}, T_{\mathsf{level}-1}^{(i)}, \sigma, \mathbf{M}_{\mathsf{level}}^{(i,b)})$

   For all $\mathsf{level} > L - j^*$, $i \leq \ell_{\mathrm{PRG}}$ and $b \in \{0,1\}$ choose $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \chi^{m \times m}$.

**Game** $(4, j^*, 3)$**:** In this experiment, the challenger chooses $\mathbf{C}^{(i,b)}_{L-j^*}$ from the noise distribution.

7. Next, it generates the components for each level. For each level $\mathsf{level} \in [1, L-j^*]$, do the following:

   (a) Choose matrices $\mathbf{S}^{(0)}_{\mathsf{level}}, \mathbf{S}^{(1)}_{\mathsf{level}} \leftarrow \chi^{n\times n}$ and $\mathbf{E}^{(i,0)}_{\mathsf{level}}, \mathbf{E}^{(i,1)}_{\mathsf{level}} \leftarrow \chi^{5n\times m}$ for $i \le \ell_{\mathrm{PRG}}$.

   (b) For $b \in \{0,1\}$, set matrix $\mathbf{D}^{(i,b)}_{\mathsf{level}}$ as a permutation of the matrix blocks of $\mathbf{B}^{(i)}_{\mathsf{level}}$ according to the permutation $\sigma^{(i)}_{\mathsf{level},b}(\cdot)$.

   (c) Set $\mathbf{M}^{(i,b)}_{\mathsf{level}} = \left(\mathbf{I}_5 \otimes \mathbf{S}^{(b)}_{\mathsf{level}}\right) \cdot \mathbf{D}^{(i,b)}_{\mathsf{level}} + \mathbf{E}^{(i,b)}_{\mathsf{level}}$ for $i \le \ell_{\mathrm{PRG}}$.

   (d) If $\mathsf{level} = L - j^*$, then choose $\mathbf{C}^{(i,b)}_{L-j^*} \leftarrow \chi^{m\times m}$ for $b \in \{0,1\}$.
   Else compute $\mathbf{C}^{(i,b)}_{\mathsf{level}} \leftarrow \mathsf{SamplePre}(\mathbf{B}^{(i)}_{\mathsf{level}-1}, T^{(i)}_{\mathsf{level}-1}, \sigma, \mathbf{M}^{(i,b)}_{\mathsf{level}})$

For all $\mathsf{level} > L - j^*$, $i \le \ell_{\mathrm{PRG}}$ and $b \in \{0,1\}$ choose $\mathbf{C}^{(i,b)}_{\mathsf{level}} \leftarrow \chi^{m\times m}$.

**Game** 5**:** In this experiment, the challenger computes an encryption of $\mathbf{0}^{|P|}$ instead of an encryption of $P$. Note that the LHE secret key $\mathsf{sk}$ is not required at any other step in this experiment. This corresponds to the challenger simulating the program.

2. It chooses $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^{d\log d})$ and sets $\mathsf{ct} \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}, \mathbf{0}^{|P|})$.

### 4.2.3 Analysis

We will now show that the above hybrid experiments are computationally indistinguishable. For any adversary $\mathcal{A}$, let $p^{\mathcal{A}}_i$ denote the probability that the adversary outputs 1 in **Game** $i$.

**Claim 4.4.** For any adversary $\mathcal{A}$, $|p^{\mathcal{A}}_0 - p^{\mathcal{A}}_1| \le \mathrm{negl}(\lambda)$.

*Proof.* The proof of this claim relies on the fact that a pairwise independent hash function (with appropriate domain and co-domain) is a strong extractor. There are two differences between these two experiments. First, in one case, the challenger chooses $h \leftarrow \mathcal{H}$, $\alpha \leftarrow \{0,1\}^{\ell_{\mathrm{out}}}$ and sets $\beta = \mathrm{PRG}(h(\alpha))$, while in the other case, the challenger chooses $h \leftarrow \mathcal{H}$, $\alpha' \leftarrow \{0,1\}^{\ell_{\mathrm{out}}}$ and sets $\beta = \mathrm{PRG}(\alpha')$. Secondly, in **Game** 0, the challenger ensures that the matrices $\mathbf{S}^{(0)}_{\mathsf{level}}, \mathbf{S}^{(1)}_{\mathsf{level}}$ are full-rank, while in **Game** 1, it chooses these matrices from $\chi$ without the rank check. Note that any $n \times n$ matrix sampled from $\chi$ is full rank with all but negligible probability. Therefore, this switch is indistinguishable. We will now show that switching $h(\alpha)$ to a uniformly random string is also statistically indistinguishable.

Suppose there exists an adversary $\mathcal{A}$ such that $|p^{\mathcal{A}}_0 - p^{\mathcal{A}}_1| = \epsilon$. We can construct a reduction algorithm $\mathcal{B}$ that can break the strong extraction property of $\mathcal{H}$ with probability $\epsilon$. The reduction algorithm receives $(h, \gamma)$ from the strong extractor challenger, and $P, \mathsf{msg}$ from the adversary $\mathcal{A}$. It chooses LHE keys $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^{d\log d})$, computes an encryption of $P$ as $\mathsf{ct} \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}, P)$ and sets $\beta = \mathrm{PRG}(\gamma)$. Next, it defines the program $Q$ which takes as input an LHE ciphertext, and has $\mathsf{sk}$ hardwired. $Q$ first decrypts the input ciphertext to compute $y$, then computes $\mathrm{PRG}(h(y))$. Let $\{\mathsf{BP}^{(i)}\}_i$ be the branching progams corresponding to $Q$. The reduction algorithm computes the components using $\mathsf{Comp\text{-}Gen}(\{\mathsf{BP}^{(i)}\}_i, \beta, \mathsf{msg})$, which is identical in both experiments. It sends the components, together with $\mathsf{ek}$ and $\mathsf{ct}$, to $\mathcal{A}$ and the adversary outputs a bit $b'$. If $b' = 1$, the reduction algorithm guesses that $\gamma$ is an evaluation of the hash function. The reduction algorithm has advantage $|p^{\mathcal{A}}_0 - p^{\mathcal{A}}_1| = \epsilon$ in the strong extractor experiment. ∎

**Claim 4.5.** Assuming the security of PRG, for any PPT adversary $\mathcal{A}$, $|p^{\mathcal{A}}_1 - p^{\mathcal{A}}_2| \le \mathrm{negl}(\lambda)$.

*Proof.* We describe a reduction algorithm $\mathcal{B}$ which plays the indistinguishability based game with PRG challenger.[12] $\mathcal{B}$ receives $\beta$ from the PRG challenger, and $P, \mathsf{msg}$ from the adversary $\mathcal{A}$. It then chooses

[12] Note that if $\mathcal{A}$ is a non-uniform attacker, then our reduction algorithm will be non-uniform as well.

$(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^{d \log d})$ and computes $\mathsf{ct} \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}, P)$. Next, it defines the program $Q$ which takes as input an LHE ciphertext, and has $\mathsf{sk}$ hardwired. $Q$ first decrypts the input ciphertext to compute $y$, then computes $\mathrm{PRG}(h(y))$. Let $\{\mathsf{BP}^{(i)}\}_i$ be the branching progams corresponding to $Q$. The reduction algorithm computes the components using $\mathsf{Comp\text{-}Gen}(\{\mathsf{BP}^{(i)}\}_i, \beta, \mathsf{msg})$, which is identical in both experiments. It sends the components, together with $\mathsf{ek}$ and $\mathsf{ct}$, and the adversary outputs a bit $b'$. If $b' = 1$, the reduction algorithm guesses that $\beta$ is an evaluation of the hash function.

Note that when the PRG challenger sends a pseudorandom string $\beta = \mathrm{PRG}(\alpha')$ for a uniformly random $\alpha'$, then $\mathcal{B}$ exactly simulates the view of $\mathsf{Game}$ 1 for $\mathcal{A}$. Otherwise if the PRG challenger outputs a uniformly random string, then $\mathcal{B}$ exactly simulates the view of $\mathsf{Game}$ 2. Therefore, if $|p_1^{\mathcal{A}} - p_2^{\mathcal{A}}|$ is non-negligible, then PRG is not a secure pseudorandom generator. ∎

**Claim 4.6.** For any adversary $\mathcal{A}$, $|p_2^{\mathcal{A}} - p_3^{\mathcal{A}}| \leq \mathrm{negl}(\lambda)$.

*Proof.* This step is information theoretic, and uses the Leftover Hash Lemma (matrix version, Corollary 2.1).

Note that the difference between $\mathsf{Game}$ 2 and 3 is the way top level matrices $\mathbf{B}_L^{(i)}$ are sampled during the component generation phase. In $\mathsf{Game}$ 2, matrix $\mathbf{B}^{(\ell_{\mathrm{PRG}})}_{L,\mathsf{st}_L^{\ell_{\mathrm{PRG}}}}$ is chosen as

$$\mathbf{B}^{(\ell_{\mathrm{PRG}})}_{L,\mathsf{st}_L^{(\ell_{\mathrm{PRG}})}} = \begin{cases} -\left(\sum_{i \leq \ell_{\mathrm{PRG}}-1 \,:\, \beta_i = 0} \mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}} + \sum_{i \leq \ell_{\mathrm{PRG}}-1 \,:\, \beta_i = 1} \mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}}\right) & \text{if } \mathsf{msg} = 0 \\ -\left(\sum_{i \leq \ell_{\mathrm{PRG}}-1 \,:\, \beta_i = 0} \mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}} + \sum_{i \leq \ell_{\mathrm{PRG}}-1 \,:\, \beta_i = 1} \mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}}\right) + \sqrt{q} \cdot \left[\mathbf{I}_n \,||\, \mathbf{0}^{n \times (m-n)}\right] & \text{if } \mathsf{msg} = 1 \end{cases}$$

where $\mathsf{st}_L^{(\ell_{\mathrm{PRG}})}$ is $\mathsf{acc}^{(\ell_{\mathrm{PRG}})}$ if $\beta_{\ell_{\mathrm{PRG}}} = 1$, and $\mathsf{rej}^{(\ell_{\mathrm{PRG}})}$ otherwise. It can be equivalently written as follows

$$\mathbf{B}^{(\ell_{\mathrm{PRG}})}_{L,\mathsf{st}_L^{(\ell_{\mathrm{PRG}})}} = -\mathbf{A} \cdot \mathbf{R}, \qquad \mathbf{A} = \left[\mathbf{B}^{(1)}_{L,\mathsf{rej}^{(1)}} \,||\, \mathbf{B}^{(1)}_{L,\mathsf{acc}^{(1)}} \,||\, \ldots \,||\, \mathbf{B}^{(\ell_{\mathrm{PRG}}-1)}_{L,\mathsf{rej}^{(\ell_{\mathrm{PRG}}-1)}} \,||\, \mathbf{B}^{(\ell_{\mathrm{PRG}}-1)}_{L,\mathsf{acc}^{(\ell_{\mathrm{PRG}}-1)}}\right]$$

where $\mathbf{R} = \mathbf{u} \otimes \mathbf{I}_m \in \mathbb{Z}_q^{2m(\ell_{\mathrm{PRG}}-1) \times m}$, $\mathbf{u} = (u_1, \ldots, u_{2\ell_{\mathrm{PRG}}-2})^\top \in \{0,1\}^{2\ell_{\mathrm{PRG}}-2}$ and for all $i \leq \ell_{\mathrm{PRG}} - 1$, $u_{2i} = \beta_i$ and $u_{2i-1} = 1 - \beta_i$. That is, matrix $\mathbf{R}$ consists of $2\ell_{\mathrm{PRG}} - 2$ submatrices where if $\beta_i = 1$, then its $2i^{th}$ submatrix is identity and $(2i-1)^{th}$ submatrix is zero, otherwise it is the opposite. Let $\mathcal{R}$ denote the distribution of matrix $\mathbf{R}$ as described above with $\beta$ drawn uniformly from $\{0,1\}^{\ell_{\mathrm{PRG}}}$. Note that $\mathbf{H}_\infty(\mathcal{R}) = \ell_{\mathrm{PRG}} - 1$ (min-entropy of $\mathcal{R}$), and $\ell_{\mathrm{PRG}} > m \cdot n \log_2 q + \omega(\log n)$. Therefore, it follows (from Corollary 2.1) that

$$\left\{\left(\mathbf{A}, \mathbf{B}^{(\ell_{\mathrm{PRG}})}_{L,\mathsf{st}_L^{(\ell_{\mathrm{PRG}})}} = -\mathbf{A} \cdot \mathbf{R}\right) \;:\; \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times 2m(\ell_{\mathrm{PRG}}-1)}, \mathbf{R} \leftarrow \mathcal{R}\right\}$$

$$\approx_s$$

$$\left\{\left(\mathbf{A}, \mathbf{B}^{(\ell_{\mathrm{PRG}})}_{L,\mathsf{st}_L^{(\ell_{\mathrm{PRG}})}}\right) \;:\; \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times 2m(\ell_{\mathrm{PRG}}-1)}, \mathbf{B}^{(\ell_{\mathrm{PRG}})}_{L,\mathsf{st}_L^{(\ell_{\mathrm{PRG}})}} \leftarrow \mathbb{Z}_q^{n \times m}\right\}$$

Thus, $|p_2^{\mathcal{A}} - p_3^{\mathcal{A}}|$ is negligible in the security parameter for all adversaries $\mathcal{A}$. ∎

**Claim 4.7.** For all $j^* \in [1, L]$, for all $\mathcal{A}$, $|p^{\mathcal{A}}_{(4,j^*-1,3)} - p^{\mathcal{A}}_{(4,j^*,0)}| \leq \mathrm{negl}(\lambda)$.

*Proof.* This proof relies on the Matrix Well Distributedness property of $\mathsf{TrapGen}$ (see Definition 2.1). The only difference in $\mathsf{Game}$ $(4, j^* - 1, 3)$ and $\mathsf{Game}$ $(4, j^*, 0)$ is the choice of $\{\mathbf{B}^{(i)}_{L-j^*}\}$. In $\mathsf{Game}$ $(4, j^* - 1, 3)$, $\{\mathbf{B}^{(i)}_{L-j^*}\}$ is chosen with a trapdoor for all $i \leq \ell_{\mathrm{PRG}}$. In $\mathsf{Game}$ $(4, j^*, 0)$, these matrices are chosen uniformly at random. Note that in both the experiments, the trapdoors corresponding to $\mathbf{B}^{(i)}_{L-j^*}$ are not used, since the matrix components $\{\mathbf{C}^{(i,b)}_{L-j^*+1}\}_{i,b}$ are chosen from the noise distribution $\chi$. As a result, using the Matrix Well Distributedness property of $\mathsf{TrapGen}$, we can argue that these two games are statistically indistinguishable, and therefore $|p^{\mathcal{A}}_{(4,j^*-1,3)} - p^{\mathcal{A}}_{(4,j^*,0)}| \leq \mathrm{negl}(\lambda)$. ∎

**Claim 4.8.** Assuming the LWE with short secrets assumption (Assumption 2) For all $j^* \in [0, L-1]$, for all PPT adversaries $\mathcal{A}$, $|p^{\mathcal{A}}_{(4,j^*,0)} - p^{\mathcal{A}}_{(4,j^*,1)}| \leq \text{negl}(\lambda)$

*Proof.* In $\mathsf{Game}\ (4, j^*, 0)$, the challenger computes $\mathbf{M}^{(i,0)}_{L-j^*}$ as $\left(\mathbf{I}_5 \otimes \mathbf{S}^{(0)}_{L-j^*}\right) \cdot \mathbf{D}^{(i,0)}_{L-j^*} + \mathbf{E}^{(i,0)}_{L-j^*}$. In $\mathsf{Game}\ (4, j^*, 1)$, this matrix is chosen uniformly at random. We will use the LWE assumption (short secrets version) to prove that these two hybrids are indistinguishable.

Suppose there exists a PPT adversary $\mathcal{A}$ that can distinguish between these two games with advantage $\epsilon$. Then we can consruct a reduction algorithm $\mathcal{B}$ that can break the $\mathsf{LWE\text{-}ss}$ assumption. First, $\mathcal{B}$ receives as $\mathsf{LWE\text{-}ss}$ challenge two matrices $(\mathbf{F}, \mathbf{G})$ of dimensions $n \times 5\ell_{\text{PRG}} \cdot m$, and $P, \mathsf{msg}$ from the adversary $\mathcal{A}$. It then partitions $\mathbf{F}$ into $\ell_{\text{PRG}}$ submatrices of dimensions $n \times 5m$ as $[\mathbf{F}^{(1)} || \ldots || \mathbf{F}^{(\ell_{\text{PRG}})}]$. Further, each matrix $\mathbf{F}^{(i)}$ is parsed into 5 matrices of dimensions $n \times m$ as $[\mathbf{F}^{(i)}_1 || \ldots || \mathbf{F}^{(i)}_5]$. Similarly, the matrix $\mathbf{G}$ is partitioned into $\{\mathbf{G}^{(i)}_k\}_{i \leq \ell_{\text{PRG}}, k \leq 5}$.

The reduction algorithm chooses the LHE keys and computes the ciphertext as in the two games. Next, it needs to choose the matrix components. First, it chooses the top $j^* - 1$ matrices $\{\mathbf{B}^{(i)}_j\}_{i \leq \ell_{\text{PRG}}, j > L-j^*}$ uniformly at random, and the top $j^* - 1$ components $\{(\mathbf{C}^{(i,0)}_j, \mathbf{C}^{(i,1)}_j)\}_{i \leq \ell_{\text{PRG}}, j > L-j^*}$ from the noise distribution. It then sets $\mathbf{B}^{(i)}_{L-j^*,k} = \mathbf{F}^{(i)}_k$ for all $i \leq \ell_{\text{PRG}}$, $k \leq 5$.

The reduction algorithm then uses $\mathbf{G}^{(i)}_k$ to define $\mathbf{M}^{(i,0)}_{L-j^*}$ as follows:

$$\mathbf{M}^{(i,0)}_{L-j^*} = \begin{bmatrix} \mathbf{G}^{(i)}_{\sigma^{(i)}_{L-j^*,0}(1)} \\ \vdots \\ \mathbf{G}^{(i)}_{\sigma^{(i)}_{L-j^*,0}(5)} \end{bmatrix}.$$

The matrix $\mathbf{M}^{(i,1)}_{L-j^*}$ is computed identically in both the games. Using $\mathbf{B}^{(i)}_{L-j^*}$, the reduction algorithm can compute $\mathbf{M}^{(i,1)}_{L-j^*}$. Finally, it computes $\mathbf{C}^{(i,b)}_{L-j^*}$ for all $i \leq \ell_{\text{PRG}}, b \in \{0,1\}$.

The remaining components are computed identically in both games, and the reduction algorithm can perfectly simulate them. Finally, it sends all the components, together with the LHE evaluation key and ciphertext. The adversary sends a bit $b'$. If $b' = 1$, the reduction algorithm guesses that $\mathbf{G} = \mathbf{S} \cdot \mathbf{F} + \mathbf{E}$ for some matrices $\mathbf{S}, \mathbf{E}$ drawn from the noise distribution. If $b' = 0$, then the reduction algorithm guesses that $\mathbf{G}$ is random.

If $\mathbf{G}$ is an LWE sample, then $\mathcal{B}$ simulates $\mathsf{Game}\ (4, j^*, 0)$, else it simulates $\mathsf{Game}\ (4, j^*, 1)$. The advantage of $\mathcal{B}$ is $p^{\mathcal{A}}_{(4,j^*,0)} - p^{\mathcal{A}}_{(4,j^*,1)}$.

■

**Claim 4.9.** Assuming the LWE with short secrets assumption (Assumption 2) For all $j^* \in [0, L-1]$, for all PPT adversaries $\mathcal{A}$, $|p^{\mathcal{A}}_{(4,j^*,1)} - p^{\mathcal{A}}_{(4,j^*,2)}| \leq \text{negl}(\lambda)$

*Proof.* In $\mathsf{Game}\ (4, j^*, 1)$, the challenger computes $\mathbf{M}^{(i,1)}_{L-j^*}$ as $\left(\mathbf{I}_5 \otimes \mathbf{S}^{(1)}_{L-j^*}\right) \cdot \mathbf{D}^{(i,1)}_{L-j^*} + \mathbf{E}^{(i,1)}_{L-j^*}$. In $\mathsf{Game}\ (4, j^*, 2)$, this matrix is chosen uniformly at random. This proof is identical to the previous proof. ■

**Claim 4.10.** For all $j^* \in [0, L-1]$, for all $\mathcal{A}$, $|p^{\mathcal{A}}_{(4,j^*,2)} - p^{\mathcal{A}}_{(4,j^*,3)}| \leq \text{negl}(\lambda)$.

*Proof.* The proof of this claim relies on the *preimage well-distributedness property* of lattice trapdoor sampler $(\mathsf{TrapGen}, \mathsf{SamplePre})$ (Definition 2.1). Note that the only difference between these two games is with regard to $\mathbf{C}^{(i,b)}_{L-j^*}$. In $\mathsf{Game}\ (4, j^*, 2)$ these matrices are computed using the $\mathsf{SamplePre}$ algorithm. In $\mathsf{Game}\ (4, j^*, 3)$, they're chosen from the noise distribution $\chi$. Since $\mathbf{M}^{(i,b)}_{L-j^*}$ is chosen uniformly at random, we can use the preimage well-distributedness property to argue that the two games are statistically indistinguishable. ■

**Claim 4.11.** Assuming the LHE scheme is IND-CPA secure, for all PPT adversaries $\mathcal{A}$, $|p^{\mathcal{A}}_{4,L,0} - p^{\mathcal{A}}_{5}| \leq \mathrm{negl}(\lambda)$.

*Proof.* The only difference between $p^{\mathcal{A}}_{4,L,0}$ and $p^{\mathcal{A}}_{5}$ is that the LHE ciphertext is an encryption of $P$ in one case, and an encryption of $\mathbf{0}^{|P|}$ in the other. Note that in both games, the LHE secret key is only used for computing the ciphertext (it is not used by the branching programs $\mathsf{BP}^{(i)}$). Suppose there exists an adversary $\mathcal{A}$ that can distinguish between these two games. Then we can construct a reduction algorithm $\mathcal{B}$ that breaks the IND-CPA security of the LHE scheme.

The reduction algorithms receives $P, \mathsf{msg}$ from the adversary $\mathcal{A}$, and sends $P, \mathbf{0}^{|P|}$ to the LHE challenger. It receives an evaluation key $\mathsf{ek}$ and a ciphertext $\mathsf{ct}$. It then chooses the matrix components $\mathbf{B}^{(i)}_0$ uniformly at random, and the components $\{\mathbf{C}^{(i,b)}_j\}_{i,j,b}$ from the noise distribution. Finally, it sends $\mathsf{ek}, \mathsf{ct}$ and the matrix components to the adversary. The adversary sends a bit $b'$. If $b' = 1$, then the reduction algorithm guesses that $\mathsf{ct}$ is an encryption of $P$, else it guesses that $\mathsf{ct}$ is an encryption of $\mathbf{0}^{|P|}$. Note that the advantage of $\mathcal{B}$ in the IND-CPA game is $|p^{\mathcal{A}}_{(4,L,0)} - p^{\mathcal{A}}_{5}|$. ∎

## 4.3 On the Need for a Pseudo Random Generator

An interesting question is whether the application of a Pseudo Random Generator is actually necessary for the security of our construction. A natural alternative construction would be to use the same overall structure with the modification that the circuit $Q$ computes $Q(\mathsf{ct}_1, \ldots, \mathsf{ct}_{\ell_{\mathrm{out}}}) = \mathsf{LHE.Dec}(\mathsf{lhe.sk}, \mathsf{ct}_1) \,||\, \cdots \,||\, \mathsf{LHE.Dec}(\mathsf{lhe.sk}, \mathsf{ct}_{\ell_{\mathrm{out}}})$. In particular, $Q$ just performs an LHE decryption and does not compute a pairwise independent hash and pseudo random generator to the result. In this case we set $\beta = \alpha$ instead of $\beta = \mathrm{PRG}(h(\alpha))$.

While we do not rule out that such a construction might work, there appear to be some inherent tensions in setting parameters to prove such a system secure. Let's return to the proof of the scheme as it is. A critical proof step is to move from Game 2 to Game 3 where in Game 2 it chooses the $\mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}}, \mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}}$ values such that

$$\sum_{i \,:\, \beta_i = 0} \mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}} + \sum_{i \,:\, \beta_i = 1} \mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}} = \begin{cases} \mathbf{0}^{n \times m} & \text{if } \mathsf{msg} = 0. \\ \sqrt{q} \cdot \left[\mathbf{I}_n \,||\, \mathbf{0}^{n \times (m-n)}\right] & \text{if } \mathsf{msg} = 1. \end{cases}$$

Whereas in Game 3 all are chosen uniformly at random. By our choice of parameters we have that $\ell_{\mathrm{PRG}} \gg n \cdot m \cdot \log q$. Therefore, with high probability, there will be many strings $w$ such that $\sum_{i \,:\, w_i = 0} \mathbf{B}^{(i)}_{L,w_i} = \mathbf{0}$. It follows that (in the case where the message bit is 0) programming the matrices so that on a random $\beta$ $\sum_{i \,:\, \beta_i = 0} \mathbf{B}^{(i)}_{L,\beta_i} = \mathbf{0}$ is statistically close to just choosing them uniformly at random.

Now if we instead use the alternative construction we have that there are $\ell_{\mathrm{out}}$ pairs of matrices instead at the last level instead of $\ell_{\mathrm{PRG}}$. In our construction $\ell_{\mathrm{out}} < \ell_{\mathrm{in}}$ since the input HE ciphertext must be big enough to encrypt $\alpha$. And $\ell_{\mathrm{in}} < \log q$ due to the noise accumulation in our construction. So we have that $\ell_{\mathrm{out}} \ll \log q$ and the statistical argument we used earlier for going from Game 2 to Game 3 no longer holds. We might instead consider a computational argument for moving between these games where the LWE assumption with binary secrets is a natural candidate. Unfortunately, known conversions [BLP+13, MP13] between standard LWE and binary LWE require the LWE challenge vector to increase by a factor of $\log q$ which would again violate our parameter constraints.

Another interesting question is whether the pairwise independent hash function could be removed from our construction where the circuit $Q$ would go straight from decrypting the HE ciphertext to evaluating a pseudorandom generator. Currently, the pairwise indepdent hash function is in place to assure semi-statistical correctness of Definition 3.3. Consider a pseudorandom generator PRG that always ignored its first input bit. Furthermore consider a program $P$, input $x$, lock value $\alpha$ such that $P(x) = \tilde{\alpha}$, where $\tilde{\alpha}$ is a bitstring that equals $\alpha$ at all bit positions except the first. Then the triple $P$, input $x$, lock value $\alpha$ would violate semi-statistical correctness under the construction where the pairwise independent hash function is removed.

At the same time if we were to use an *injective* pseudo random generator such issues are not a factor and the construction will actually achieve statistical correctness of Definition 3.2. We show this formally in Appendix D.

# 5 Predicate Encryption: Achieving 1-Sided Security

Below we first give a formal description of our transformation for the case of a key-policy ABE (KP-ABE) system where the decryption algorithms are of a bounded depth and size. In Appendix E we show a slight generalization of the first transformation where we add a level of indirection using fully homomorphic encryption to remove the depth and size constraints. We follow (in this section) by enumerating different forms of Attribute-Based Encryption and informally discussing how our transformation could be adapted to each case. We finally sketch a derivative application where we can encrypt to an arbitrarily formed public key and hide from everyone except the receiver which public key we encrypted to.

## 5.1 Key-Policy ABE with Bounded Decryption Depth and Size

In this section, we construct a PE scheme which achieves 1-sided security from any KP-ABE scheme and a lockable obfuscator. Our construction inherits the attribute space and predicate class of the underlying KP-ABE scheme.

**Outline.** The central idea is to hide the ciphertext inside an obfuscated circuit. During encryption, instead of encrypting the message $m$ (using the underlying encryption algorithm), the encryptor chooses a lock value $\alpha$ for obfuscation and encrypts the lock value $\alpha$ under the input attribute $x$ to get ciphertext $\mathsf{ct}_\alpha$. Next, it obfuscates (under the same lock value $\alpha$) the decryption circuit of the underlying ABE scheme with the ciphertext $\mathsf{ct}_\alpha$ (encrypting lock value) hardwired. Now, output of this obfuscated circuit $\widetilde{P}$ would be set to be the actual message $m$ and the obfuscated circuit $\widetilde{P}$ will be the final ciphertext. Observe that given any valid secret key $\mathsf{sk}_C$ (such that $C(x) = 1$), one could simply evaluate the obfuscated circuit using the secret key and its output would guaranteed to be message $m$ by correctness of the underlying decryption and evaluation algorithms. At a high level, this hides the attribute $x$ because if the adversary can not decrypt the ciphertext $\mathsf{ct}_\alpha$ containing the lock value $\alpha$ (i.e, it does not know the secret key for the corresponding attribute), then it has not no information about the lock value $\alpha$. Therefore, by the obfuscation security, we can guarantee that the circuit $\widetilde{P}$ (i.e., challenge ciphertext) could be simulated instead. Below we describe our construction $\mathcal{PE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ for attribute spaces $\{\mathcal{X}_\lambda\}_\lambda$, predicate classes $\{\mathcal{C}_\lambda\}_\lambda$ and 1-bit messages.

**Construction.** Let $\mathcal{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.Enc}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Dec})$ be a key-policy attribute based encryption scheme for set of attribute spaces $\{\mathcal{X}_\lambda\}_\lambda$, predicate classes $\{\mathcal{C}_\lambda\}_\lambda$ and message spaces $\{\mathcal{M}_\lambda\}_\lambda$ with decryption circuit of depth $d(\lambda)$ and secret key space $\{0,1\}^{\ell(\lambda)}$. Also, let $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ be a lockable obfuscator for circuit class $\mathcal{C}_{\ell,k,d}$ (i.e., the class of depth $d(\lambda)$ circuits with $\ell(\lambda)$ bit input and $k(\lambda)$ bit output). For simplicity, assume that $\mathcal{M}_\lambda = \{0,1\}^{k(\lambda)}$. Below we describe our construction. For notational convenience, let $\ell = \ell(\lambda), k = k(\lambda)$ and $d = d(\lambda)$.

- $\mathsf{Setup}(1^\lambda, \mathcal{X}_\lambda, \mathcal{C}_\lambda) \to (\mathsf{pp}, \mathsf{msk})$. The setup algorithm runs $\mathsf{ABE.Setup}$ to generate public parameters and master secret key as $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda, \mathcal{X}_\lambda, \mathcal{C}_\lambda, 1^k)$.
- $\mathsf{Enc}(\mathsf{pp}, x, m) \to \mathsf{ct}$. The encryption algorithm chooses a random string $\alpha \leftarrow \{0,1\}^k$ and encrypts it as $\mathsf{ct}_\alpha \leftarrow \mathsf{ABE.Enc}(\mathsf{pp}, x, \alpha)$. Next, it obfuscates the decryption circuit with message $m$ and lock $\alpha$ as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{ABE.Dec}(\cdot, \mathsf{ct}_\alpha), m, \alpha)$. Finally, it outputs the ciphertext as $\mathsf{ct} = \widetilde{P}$.
- $\mathsf{KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$. The key generation algorithm runs $\mathsf{ABE.KeyGen}$ to generate the secret key as $\mathsf{sk}_C \leftarrow \mathsf{ABE.KeyGen}(\mathsf{msk}, C)$.

- $\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}) \to m$ or $\perp$ . Let $\mathsf{ct} = \widetilde{P}$. The decryption algorithm evaluates the obfuscated program on input $\mathsf{sk}_C$, and outputs $\mathsf{Eval}(\widetilde{P}, \mathsf{sk}_C)$.

**Remark 5.1** (Preservation of Algorithms and Retroactive Application)**.** We would like to point out that the above transformation preserves the original ABE setup and key generation algorithms as well as the private key structure. This could allow some flexibility in encryption. If the user encrypting data feels that hiding the attributes is important, he can apply our transformation. If not, he can default to the original scheme. Also, it allows for one-sided predicate encryption to be retroactively applied to an ABE scheme that has already published its public parameters and distributed its keys. Of course, only newly generated ciphertexts generated with using the transformation would have the attributes hidden.

**Correctness.** For all $\lambda \in \mathbb{N}$, message $m \in \{0,1\}$, attribute $x \in \mathcal{X}_\lambda$, public parameters and master secret key $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda, \mathcal{X}_\lambda, \mathcal{C}_\lambda, \mathcal{M}_\lambda)$, the ciphertext corresponding to message $m$ under attribute $x$ in our scheme is an obfuscated circuit $\widetilde{P}$, where $\alpha \leftarrow \{0,1\}^k$, $\mathsf{ct}_\alpha \leftarrow \mathsf{ABE.Enc}(\mathsf{pp}, x, \alpha)$ and $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{ABE.Dec}(\cdot, \mathsf{ct}_\alpha), m, \alpha)$.

For any predicate $C \in \mathcal{C}_\lambda$, the corresponding secret key in our scheme is simply $\mathsf{sk}_C \leftarrow \mathsf{ABE.KeyGen}(\mathsf{msk}, C)$. Consider the following two cases:

1. $C(x) = 1$ : We know that if $C(x) = 1$, then with all but negligible probability $\mathsf{ABE.Dec}(\mathsf{sk}_C, \mathsf{ct}_\alpha) = \alpha$. This follows from correctness of the $\mathcal{ABE}$ scheme. Next, using correctness of obfuscation, we can conclude that $\mathsf{Eval}(\widetilde{P}, \mathsf{sk}_C) = m$ as $\mathsf{ABE.Dec}(\mathsf{sk}_C, \mathsf{ct}_\alpha) = \alpha$. Therefore, if $C(x) = 1$, then $\Pr[\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct} = \widetilde{P}) = m] \geq 1 - \mathrm{negl}(\lambda)$.
2. $C(x) = 0$ : Similarly, we know that if $C(x) = 0$, then with all but negligible probability $\mathsf{ABE.Dec}(\mathsf{sk}_C, \mathsf{ct}_\alpha) = \perp$. Also by correctness of obfuscation, we can conclude that if $\mathsf{ABE.Dec}(\mathsf{sk}_C, \mathsf{ct}_\alpha) \neq \alpha$, then $\Pr[\mathsf{Eval}(\widetilde{P}, \mathsf{sk}_C) = \perp] \geq 1 - \mathrm{negl}(\lambda)$. Since $\perp \neq \alpha$, therefore combining these two facts we know that if $C(x) = 0$, then $\Pr[\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct} = \widetilde{P}) = \perp] \geq 1 - \mathrm{negl}'(\lambda)$ for some negligible function $\mathrm{negl}'(\cdot)$.

Therefore, $\mathcal{PE}$ satisfies the predicate encryption correctness condition.

**Security.** We will now show that the scheme described above achieves 1-sided security as per Definition A.2. Formally, we prove the following.

**Theorem 5.1.** If $\mathcal{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.Enc}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Dec})$ is a fully secure attribute based encryption for set of attribute spaces $\{\mathcal{X}_\lambda\}_\lambda$, predicate classes $\{\mathcal{C}_\lambda\}_\lambda$ and message spaces $\{\mathcal{M}_\lambda\}_\lambda$ satisfying Definition A.1, and $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ is a lockable obfuscator for 1-bit messages and circuit class $\{\mathcal{C}_{\ell,k,d}\}_\lambda$ satisfying Definition 3.4, then $\mathcal{PE}$ is a secure predicate encryption scheme satisfying 1-sided security as per Definition A.2 for 1-bit messages and same attribute space and predicate class as the ABE scheme.

Our proof proceeds via a sequence of hybrid games. Each game is played between the challenger and attacker $\mathcal{A}$. Let $\mathcal{A}$ be any PPT adversary that wins the 1-sided security game with non-negligible advantage. We argue that such an adversary must break security of at least one underlying primitive. The first game corresponds to the 1-sided security game as described in Definition A.2. In the final game, the challenge ciphertext is simulated and does not contain any information about the challenge messages.

We will first define the sequence of hybrid games, and then show that they are computationally indistinguishable.

**Game 1:** In this game, the challenge ciphertext $\mathsf{ct}$ is honestly generated, i.e. by first choosing a uniformly random lock $\alpha$, then encrypting it under attribute $x$ to get ciphertext $\mathsf{ct}_\alpha$ and finally obfuscating the decryption circuit with lock $\alpha$ and message $m_b$.

1. **Setup Phase.** The challenger sets up by generating public parameters and master secret key as $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda, \mathcal{X}_\lambda, \mathcal{C}_\lambda, 1^k)$. It sends $\mathsf{pp}$ to $\mathcal{A}$.

2. **Pre-Challenge Query Phase.** $\mathcal{A}$ queries the challenger on polynomially many predicate circuits $C_i$ to receive the corresponding secret keys as $\mathsf{sk}_{C_i} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{msk}, C_i)$.

3. **Challenge.** Next, $\mathcal{A}$ sends the challenge messages and attributes $((m_0, x_0), (m_1, x_1))$ to the challenger such that $C_i(x_0) = C_i(x_1) = 0$ for all queried predicates. The challenger chooses a random bit $b \leftarrow \{0,1\}$ and a uniformly random lock string $\alpha \leftarrow \{0,1\}^k$. It computes ciphertext $\mathsf{ct}$ as $\mathsf{ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{pp}, x_b, \alpha)$. It also generates obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{ABE.Dec}(\cdot, \mathsf{ct}), m_b, \alpha)$, and sends $\widetilde{P}$ to $\mathcal{A}$.

4. **Post-Challenge Query Phase.** $\mathcal{A}$ queries the challenger on polynomially many predicate circuits $C_i$ as before. The challenger handles these as in pre-challenge query phase.

5. **Guess.** $\mathcal{A}$ outputs it guess $b'$ and wins if $b = b'$.

**Game 2:** This is same as Game 1, except the challenger computes ciphertext $\mathsf{ct}$ as an encryption of an all zero string instead of $\alpha$.

3. **Challenge.** Next, $\mathcal{A}$ sends the challenge messages and attributes $((m_0, x_0), (m_1, x_1))$ to the challenger such that $C_i(x_0) = C_i(x_1) = 0$ for all queried predicates. The challenger chooses a random bit $b \leftarrow \{0,1\}$ and a uniformly random lock string $\alpha \leftarrow \{0,1\}^k$. It computes ciphertext $\mathsf{ct}$ as $\mathsf{ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{pp}, x_b, 0^k)$. It also generates obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{ABE.Dec}(\cdot, \mathsf{ct}), m_b, \alpha)$, and sends $\widetilde{P}$ to $\mathcal{A}$.

**Game 3:** This is same as Game 2, except the challenger does not choose the lock $\alpha$ anymore and it simulates the obfuscated program $\widetilde{P}$ instead of generating it honestly as an obfuscation of the ABE decryption circuit.

3. **Challenge.** Next, $\mathcal{A}$ sends the challenge messages and attributes $((m_0, x_0), (m_1, x_1))$ to the challenger such that $C_i(x_0) = C_i(x_1) = 0$ for all queried predicates. The challenger chooses a random bit $b \leftarrow \{0,1\}$. It generates obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathcal{O}.\mathsf{Sim}(1^\lambda, 1^s, 1)$ (where $s = |\mathsf{ABE.Dec}(\cdot, \mathsf{ct})|$), and sends $\widetilde{P}$ to $\mathcal{A}$.

Let $\mathsf{Adv}^i_{\mathcal{A}} = |\Pr[b' = b] - 1/2|$ denote the advantage of adversary $\mathcal{A}$ in guessing the bit $b$ in Game $i$. First, note that $\mathsf{Adv}^3_{\mathcal{A}} = 0$. In other words, $\mathcal{A}$'s advantage in Game 3 is 0. This is because the challenge ciphertext $\widetilde{P}$ does not contain any information about bit $b$.

To complete the proof, we establish via a sequence of lemmas that no PPT adversary $\mathcal{A}$ can distinguish between each adjacent game with non-negligible probability. We show via a sequence of lemmas that $|\mathsf{Adv}^i_{\mathcal{A}} - \mathsf{Adv}^{i+1}_{\mathcal{A}}|$ is negligible for all $i = 1, 2$. Below we discuss our lemmas in detail.

**Lemma 5.1.** If $\mathcal{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.Enc}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Dec})$ is a fully secure attribute based encryption, then for all PPT adversaries $\mathcal{A}$, $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is negligible in the security parameter $\lambda$.

*Proof.* Suppose there exists an adversary $\mathcal{A}$ such that $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is non-negligible. We construct an algorithm $\mathcal{B}$ that can distinguish encryptions of lock $\alpha$ and an all zeros string, therefore break security of the ABE scheme.

The ABE challenger generates a key pair $(\mathsf{pp}, \mathsf{msk})$ and sends $\mathsf{pp}$ to $\mathcal{B}$. $\mathcal{B}$ simply forwards the public parameters $\mathsf{pp}$ to adversary $\mathcal{A}$. $\mathcal{A}$ queries the reduction algorithm $\mathcal{B}$ on polynomially many predicates $C_i$ for corresponding secret keys. $\mathcal{B}$ forwards each predicate query $C_i$ to the ABE challenger and receives back secret key $\mathsf{sk}_{C_i}$, which it then sends to $\mathcal{A}$ as its response. Next, $\mathcal{A}$ sends the challenge message-attribute pairs $((m_0, x_0), (m_1, x_1))$ to $\mathcal{B}$ such that $C_i(x_0) = C_i(x_1) = 0$ for all queried predicates. $\mathcal{B}$ chooses a random string $\alpha \leftarrow \{0,1\}^k$ and bit $b \leftarrow \{0,1\}$, and sends $t_0 = \alpha$ and $t_1 = 0^k$ as its challenge messages and $x_b$ as the challenge attribute to the ABE challenger. The ABE challenger chooses a random bit $\beta \leftarrow \{0,1\}$, computes the challenge ciphertext $\mathsf{ct}^* \leftarrow \mathsf{ABE.Enc}(\mathsf{pp}, x_b, t_\beta)$ and sends $\mathsf{ct}^*$ to $\mathcal{B}$. After receiving $\mathsf{ct}^*$ from the challenger, $\mathcal{B}$ runs Step 3 as in Game 1. That is, $\mathcal{B}$ generates obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{ABE.Dec}(\cdot, \mathsf{ct}^*), m_b, \alpha)$,

and sends $\widetilde{P}$ to $\mathcal{A}$. Next, $\mathcal{A}$ makes more key queries and $\mathcal{B}$ answers them as before by forwarding those to ABE challenger. Finally, $\mathcal{A}$ outputs its guess $b'$. If $b = b'$, then $\mathcal{B}$ sends 0 as its guess (i.e., $\alpha$ was encrypted), otherwise it sends 1 as its guess (i.e., $0^k$ was encrypted) to the ABE challenger.

Note that if the ABE challenger encrypted $\alpha$ (i.e., $\beta = 0$), then $\mathcal{B}$ perfectly simulates Game 1 for adversary $\mathcal{A}$. Otherwise it simulates Game 2 for $\mathcal{A}$. As a result, if $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is non-negligible, then $\mathcal{B}$ breaks the ABE scheme's security with non-negligible advantage. ■

**Lemma 5.2.** If $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ is a secure lockable obfuscator, then for all PPT adversaries $\mathcal{A}$, $|\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}}|$ is negligible in the security parameter $\lambda$.

*Proof.* Suppose there exists an adversary $\mathcal{A}$ such that $|\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}}|$ is non-negligible. We construct an algorithm $\mathcal{B}$ that can distinguish an obfuscation of the ABE decryption circuit with a random lock from a simulated obfuscated program, therefore break security of the obfuscation scheme.

$\mathcal{B}$ samples an ABE key pair $(\mathsf{pp}, \mathsf{msk})$ and sends $\mathsf{pp}$ to adversary $\mathcal{A}$. $\mathcal{A}$ queries the reduction algorithm $\mathcal{B}$ on polynomially many predicates $C_i$ for corresponding secret keys. $\mathcal{B}$ answers each query with secret key $\mathsf{sk}_{C_i}$, with $\mathsf{sk}_{C_i} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{msk}, C_i)$. Next, $\mathcal{A}$ sends the challenge message-attribute pairs $((m_0, x_0), (m_1, x_1))$ to $\mathcal{B}$ such that $C_i(x_0) = C_i(x_1) = 0$ for all queried predicates. $\mathcal{B}$ chooses a random bit $b \leftarrow \{0, 1\}$ and computes ciphertext $\mathsf{ct}$ as $\mathsf{ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{pp}, x_b, 0^k)$. $\mathcal{B}$ sends the $\mathsf{ABE.Dec}(\cdot, \mathsf{ct})$ circuit along with message $m_b$ to the obfuscation challenger, and receives an obfuscated program $\widetilde{P}$. $\mathcal{B}$ finally sends $\widetilde{P}$ to $\mathcal{A}$ as the challenge ciphertext. Next, $\mathcal{A}$ makes more key queries and $\mathcal{B}$ answers them as before. Finally, $\mathcal{A}$ outputs its guess $b'$. If $b = b'$, then $\mathcal{B}$ sends 0 as its guess (i.e., ABE decryption circuit was obfuscated), otherwise it sends 1 as its guess (i.e., program $\widetilde{P}$ was simulated) as its guess to the obfuscation challenger.

Note that if the obfuscation challenger obfuscated $\mathsf{ABE.Dec}(\cdot, \mathsf{ct})$ for some lock $\alpha$, then $\mathcal{B}$ perfectly simulates Game 2 for adversary $\mathcal{A}$. Otherwise it simulates Game 3 for $\mathcal{A}$. As a result, if $|\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}}|$ is non-negligible, then $\mathcal{B}$ breaks the obfuscation scheme's security with non-negligible advantage. ■

## 5.2 Generalizing to Other Types

We now briefly discuss other forms of Attribute-Based Encryption and informally describe how our transformation could be adapted to them.

**Key-Policy ABE with Unbounded Decryption Depth.** The above construction above applies to ABE systems where there is a system wide bound on the depth of the decryption circuit and bounded key size. This can account for both for ABE schemes where there is already such a bound in place such as circuit ABE schemes from LWE or be applied to ABE schemes where no such bound is in place, but one is created for purposes of the transformation. The advantage of this is that the only added assumption is the Learning with Error from lockable obfuscation.

However, in many cases we will want to use systems that don't have such a bound. For example, most key-policy ABE systems that support keys as arbitrary boolean formulas could have decryption circuits of arbitrary depth. Note that bilinear group ABE systems are the only ones that currently support dual system encryption and adaptive security using only polynomial-hardness assumptions [Wat09, LW10, LOS+10].

To accommodate this we modify the transformation by adding an extra FHE layer as follows. The encryption algorithm starts as before by choosing a random lock value $\alpha$ and encrypting the lock value under attribute string $x$ using ABE encryption algorithm. However, it next creates a fresh FHE public-private key pair and encrypts the ABE ciphertext under the FHE public key. Next, it creates a lockable obfuscation that is created with lock value $\alpha$, the message we want to encrypt and a program that takes as input an FHE ciphertext and decrypts using the FHE secret key. The ciphertext consists of the FHE public key, ciphertext and (lockable) obfuscated program. To decrypt simply use the fully homomorphic decryption algorithm to decrypt the ABE ciphertext inside the FHE ciphertext and then submit the resulting ciphertext as input to the obfuscated program.

The security argument is almost the same as before. In the first step ABE security is used to change the ciphertext encrypting $\alpha$ to encrypt the all 0's string. Next the obfuscated program is changed to a simulated program which looses knowledge of the FHE secret key. Then we can finally change the FHE ciphertext encrypting the ABE ciphertext to encrypt the all 0's string. The transformation is provided in Appendix E.

Such a technique will be applicable to many forms of key-policy Attribute-Based encryption including those that use dual system encryption for adaptive security [Wat09, LOS+10], ABE for boolean formulas [GPSW06] and ABE for deterministic finite automata [Wat12, Att14].

**Unbounded Size, but Bounded Depth.** An in-between variant of our first transformation and the methods described above would be to accommodate ABE systems that inherently have decryption circuits that have bounded depth, but not bounded size. A prototypical example is an ABE for circuits scheme from LWE [GVW13, BGG+14]. In this case we can do the same construction as above, except use a leveled homomorphic encryption scheme where the level is chosen to match the maximum ABE circuit decryption depth. The advantage of this approach is that realizations of leveled homomorphic encryption are known from the LWE assumption [BV11, BGV12].

**Ciphertext Policy Attribute-Based Encryption.** An alternate form of Attribute-Based Encryption is ciphertext-policy ABE. In this setting the semantics associated with the keys and ciphertexts get flipped. Here a function $f$ is associated with a ciphertext and an attribute string $x$ with a private key. A user with a private key for $x$ should be able to decrypt a ciphertext associated with $f$ if and only if $f(x) = 1$. In the setting of one-sided predicate encryption we want to hide the function descriptor $f$ associated with the ciphertext from all those users who cannot decrypt it. Using universal circuits (or formulas) one can build [GJPS08] a ciphertext policy-ABE schemes from key-policy ABE scheme, however, this only works for circuits of aprori bounded size.

Using a variant of our approach from Section 5.1 we can handle all ABE schemes that have ciphertexts of bounded size. If we add a layer of fully homomorphic encryption the bounded size restriction goes away and we can handle additional schemes including those that are in the multi-authority setting [Cha07, CC09, LW11] or have ciphertext functions associated with deterministic finitie automatas [Wat12, Att14].

### 5.3 Encrypting to a Hidden Public Key

Our final application of this section is that using lockable encryption one can create a ciphertext that encrypts to a public key, but hides the identity of the public key from anyone who does not have the secret key. Such functionality was considered previously, see for example [WFS03, BBDP01, ABN08] and the references therein. However, in previous cases the hiding would only be among all public keys that shared some common parameter such as all using the same elliptic curve group. Suppose instead that we wanted to encrypt in a way that it is not clear whether it was to someone with a 4096 bit RSA key, a 2048 bit RSA key or using an ECC key.

To do this the encryption algorithm first creates a ciphertext $\mathsf{ct}$ that encrypts a randomly chosen lock value $\alpha$ using encryption under the recipients public key cryptosystem. Then the encryption algorithm creates a program $P$ that takes as input a decryption circuit description and applies it to the ciphertext $\mathsf{ct}$. The final ciphertext is a lockable obfuscation of this program $P$ under the lock $\alpha$ and the message to be transmitted. The decryption circuit description and program size should be padded to a include all decryption circuits of a chosen class. To attempt to decrypt such a ciphertext, one must first translate their secret key and decryption algorithm into a decryption circuit and then input this circuit description into the obfuscated program.

## 6 Separating IND-CPA security and Circular Security

A group of $n$ public keys and ciphertexts is said to form a key cycle [CL01] if the $i^{th}$ ciphertext encrypts the $(i+1)^{th}$ secret key. A system is said to be $n$-circular secure if key cycles of length $n$ are indistinguishable

from the case where the ciphertexts consists of encryptions to the all zeros message. Achieving circular security is very desirable in some cases. For instance, a leveled homoromorphic encryption scheme that is circular secure implies the existence of a fully homomorphic encryption scheme via Gentry's [Gen09] bootstrapping technique. As long as the levels of the homomorphic decryption circuit fit within the levels given by evaluation.

There have been several recent works that have shown that circular security, unfortunately, does not come for free. In particular, there exists IND-CPA schemes (in various contexts) that are not circular secure. Achieving such results initially depended on the existence of indistinguishability obfuscation. Subsequent works showed how in some scenarios one could make the desirable trade of assuming learning with errors instead of indistinguishability obfuscation. However, each of these constructions required the introduction of specialized LWE oriented techniques. Here we show that these separation results can instead be achieved by lockable obfuscation. The insight is simply that the original results used iO on programs that were "lockable obfuscation compatible" to begin with. So all we need to do is make very minor tweaks to the programs and then essentially copy the results. (We work things out fully below for completeness.)

The benefits are two fold. First, we are able to achieve some separation results that were not previously known under standard assumptions. And second, the solutions provided via lockable obfuscation are simpler to describe and understand in comparison to direct number theoretic constructions. Of course, direct constructions will likely be more practically efficient, although the importance of this is not clear in the setting of separation results. We now overview our three solutions along with the preceding work in each case. We present our results using the language of cycle testers as introduced by Bishop, Hohenberger and Waters [BHW15].

**Cycle testers of length $n$.** Acar et al. [ABBC10] using the SXDH assumption in bilinear groups provided an IND-CPA scheme that was not two circular secure. Subsequently, Koppula, Ramchen and Waters [KRW15] showed separation results for any $n$ cycle length assuming indistinguishability obfuscation. (Concurrently, Marcedone and Orlandi [MO14] did this from VBB.) After these works Bishop, Hohenberger and Waters [BHW15] gave a cycle tester under the LWE assumption for $n = 2$. Finally, Koppula and Waters [KW16] and Alamati and Peikert [AP16] gave cycle testers for any a priori bounded length under the learning with errors and ring learning with errors problem.[13] Below we will provide a translation of the KRW $n$ cycle testers.

**Cycle tester for bit encryption schemes.** The above separation results do not apply for when the underlying encryption scheme encrypts single bit messages. Rothblum [Rot13] first showed a separation for bit encryption cycles of length one using a high degree and non-noisy multilinear multilinear paper. Koppula, Ramchen and Waters [KRW15] (in a different solution from above) showed the separation using indistinguishability obfuscation. Very recently, Goyal, Koppula and Waters [GKW17b] gave a bit encryption cycle tester secure under the Learning with Errors assumption. Their cycle tester, however, only worked for secret key encryption while the others were public key. Below we provide a translation of the KRW bit encryption cycle tester.

**Cycle testers of unbounded length.** Returning to the case of encryption schemes with multi-bit messages Goyal, Koppula and Waters [GKW17a] recently provided a cycle tester that had no apriori bound on cycle length. Their solution requires both indistinguishability obfuscation and leveled (bootstrappable) homomorphic encryption. We show how we can swap out indistinguishability obfuscation for lockable obfuscation.

[13] In their ring LWE solution Alamati and Peikert [AP16] built testers for any cycle length that could depend on the security parameter. In their LWE solution the cycle length had to be constant.

**Program $\mathsf{P}_{\overline{\mathsf{sk}}}$**

**Constants**: Secret key $\overline{\mathsf{sk}}$.
**Inputs:** Ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$.

1. Let $\mathsf{sk}_1 = \overline{\mathsf{sk}}$. For $i = 1$ to $n$:
   (a) Decrypt $\mathsf{ct}_i$ as $(\mathsf{sk}_{(i \bmod n)+1}, \alpha_{(i \bmod n)+1}) = \overline{\mathsf{Dec}}(\mathsf{sk}_i, \mathsf{ct}_i)$.
2. If any decryption step fails, output $\perp$. Otherwise, output $\alpha_1$.

Figure 3: $n$-Cycle Tester

## 6.1 Separating IND-CPA Security from $n$-Circular Security

In this section, we construct a key cycle tester $\Gamma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Test})$ from any public key encryption scheme and a lockable obfuscator.

Let $\overline{\mathsf{E}} = (\overline{\mathsf{Setup}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ be a public key encryption scheme for message space $\{0,1\}^{2k(\lambda)}$ with decryption circuit of depth $d(\lambda)$, secret key space $\{0,1\}^{k(\lambda)}$ and ciphertext space $\{0,1\}^{\ell(\lambda)}$. Also, let $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ be a lockable obfuscator for 1-bit messages and circuit class $\mathcal{C}_{n\cdot\ell,k,n\cdot d}$ (i.e., the class of depth $n \cdot d(\lambda)$ circuits with $n \cdot \ell(\lambda)$ bit input and $k(\lambda)$ bit output). Below we describe our construction. For notational convenience, let $k = k(\lambda)$.

- $\mathsf{Setup}(1^\lambda, 1^n)$ : The setup algorithm outputs public parameters as $\mathsf{pp} = (\lambda, n)$.
- $\mathsf{KeyGen}(\mathsf{pp})$ : Let $\mathsf{pp} = (\lambda, n)$. The key generation algorithm runs $\overline{\mathsf{Setup}}$ to obtain a public-secret key pair as $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{Setup}}(1^\lambda)$. It also chooses a string $\alpha$ uniformly at random as $\alpha \leftarrow \{0,1\}^k$.

  Next, consider the program $\mathsf{P}_{\overline{\mathsf{sk}}}$ described in Figure 3. It obfuscates program $\mathsf{P}_{\overline{\mathsf{sk}}}$ with message 1 and lock $\alpha$ as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{P}_{\overline{\mathsf{sk}}}, 1, \alpha)$. Finally, it outputs the key pair as $\mathsf{pk} = (\overline{\mathsf{pk}}, \widetilde{P}), \mathsf{sk} = (\overline{\mathsf{sk}}, \alpha)$.
- $\mathsf{Enc}(\mathsf{pk}, m)$ : Let $\mathsf{pk} = (\overline{\mathsf{pk}}, \widetilde{P})$. The encryption algorithm computes ciphertext as $\mathsf{ct} \leftarrow \overline{\mathsf{Enc}}(\overline{\mathsf{pk}}, m)$.
- $\mathsf{Test}(\mathbf{pk}, \mathbf{ct})$ : Let $\mathbf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ and $\mathsf{pk}_1 = (\overline{\mathsf{pk}}_1, \widetilde{P}_1)$. The testing algorithm evaluates the obfuscated program $\widetilde{P}_1$ on input $\mathbf{ct}$ as $b = \mathsf{Eval}(\widetilde{P}_1, \mathbf{ct})$. If $b = \perp$, it outputs 0. Otherwise, it outputs $b$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ : Let $\mathsf{sk} = (\overline{\mathsf{sk}}, \alpha)$. The decryption algorithm outputs $\overline{\mathsf{Dec}}(\overline{\mathsf{sk}}, \mathsf{ct})$.[14]

**Correctness.** For all $\lambda, n \in \mathbb{N}$, public-secret key pairs $(\overline{\mathsf{pk}}_i, \overline{\mathsf{sk}}_i) \leftarrow \overline{\mathsf{Setup}}(1^\lambda)$ and strings $\alpha_i \leftarrow \{0,1\}^k$ (for $i \le n$), a key cycle of length $n$ consists of a sequence of $n$ ciphertexts $\mathbf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_n)$, where each ciphertext $\mathsf{ct}_i$ is computed as $\mathsf{ct}_i \leftarrow \overline{\mathsf{Enc}}\left(\overline{\mathsf{pk}}_i, (\overline{\mathsf{sk}}_{(i \bmod n)+1}, \alpha_{(i \bmod n)+1})\right)$. Also, the program $\widetilde{P}_1$ (which is part of public key $\mathsf{pk}_1$) is computed as $\widetilde{P}_1 \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{P}_{\overline{\mathsf{sk}}_1}, 1, \alpha_1)$.

Let $\widetilde{\mathbf{ct}} = (\widetilde{\mathsf{ct}_1}, \ldots, \widetilde{\mathsf{ct}_n})$ be encryption of zero strings, where each ciphertext $\widetilde{\mathsf{ct}_i}$ is computed as $\widetilde{\mathsf{ct}_i} \leftarrow \overline{\mathsf{Enc}}(\overline{\mathsf{pk}}_i, 0^{2k})$. For correctness of $\mathsf{Test}$ algorithm, we want to argue that it's advantage in distinguishing a sequence of encryptions of secret keys from encryptions of zeros is non-negligible.

First, note that $\mathsf{P}_{\overline{\mathsf{sk}}_1}(\mathbf{ct}) = \alpha_1$ (by construction). Therefore, by correctness of obfuscation $\mathsf{Eval}(\widetilde{P}_1, \mathbf{ct}) = 1$. Similarly, $\mathsf{P}_{\overline{\mathsf{sk}}_1}(\widetilde{\mathbf{ct}}) \neq \alpha_1$ (with all but negligible probability) as $\alpha_1$ is chosen uniformly at random and $\mathsf{P}_{\overline{\mathsf{sk}}_1}$ is independent of string $\alpha_1$. This suggests that $\Pr[\mathsf{Eval}(\widetilde{P}_1, \widetilde{\mathbf{ct}}) = \perp] \ge 1 - \mathsf{negl}(\lambda)$.

Therefore, we can conclude that $\Gamma$ satisfies cycle tester correctness condition as it can distinguish key cycles and all zero encryption with non-negligible probability.

[14] We would like to point out that cycle tester framework does not contain a decryption algorithm. However, since it is naturally available in our scheme, therefore we also specify it.

**Security.** We will now show that the scheme described above achieves IND-CPA security as per Definition 2.4. Formally, we prove the following.

**Theorem 6.1.** If $\overline{\mathsf{E}} = (\overline{\mathsf{Setup}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ is an IND-CPA secure public key encryption scheme for message space $\{0,1\}^{2k(\lambda)}$, secret key space $\{0,1\}^{k(\lambda)}$ and ciphertext space $\{0,1\}^{\ell(\lambda)}$ satisfying Definition 2.4, and $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ is a lockable obfuscator for 1-bit messages and circuit class $\mathcal{C}_{n\cdot\ell,k,d}$ satisfying Definition 3.4, then $\Gamma$ is a secure cycle tester scheme satisfying IND-CPA security as per Definition 2.4.

*Proof.* Our proof proceeds via a sequence of hybrid games. Let $\mathcal{A}$ be any PPT adversary that wins the IND-CPA security game against $\Gamma$ with non-negligible advantage. We argue that such an adversary must break security of at least one underlying primitive.

We will first define the sequence of hybrid games, and then show that they are computationally indistinguishable.

**Game 1:** This game is the original IND-CPA security game described in Definition 2.4.

1. The challenger generates a public-secret key pair as $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{Setup}}(1^\lambda)$. It uniformly samples $\alpha \leftarrow \{0,1\}^k$, and computes the obfuscation of program $\mathsf{P}_{\overline{\mathsf{sk}}}$ (described in Figure 3) as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{P}_{\overline{\mathsf{sk}}}, 1, \alpha)$. It sends the public key $\mathsf{pk} = (\overline{\mathsf{pk}}, \widetilde{P})$ to $\mathcal{A}$.
2. $\mathcal{A}$ receives $\mathsf{pk}$ from the challenger, and computes messages $m_0, m_1$. It sends $(m_0, m_1)$ as its challenge messages to the challenger.
3. The challenger chooses bit $b \leftarrow \{0,1\}$, computes $\mathsf{ct}^* \leftarrow \overline{\mathsf{Enc}}(\overline{\mathsf{pk}}, m_b)$, and sends $\mathsf{ct}^*$ to $\mathcal{A}$.
4. $\mathcal{A}$ receives challenge ciphertext $\mathsf{ct}^*$ from the challenger, and outputs its guess $b'$.
5. $\mathcal{A}$ wins if it guesses correctly, that is if $b = b'$.

**Game 2:** This game is same as Game 1, except the challenger does not choose the lock $\alpha$ anymore and it simulates the obfuscated program $\widetilde{P}$ instead of generating it honestly as an obfuscation of program $\mathsf{P}_{\overline{\mathsf{sk}}}$.

1. The challenger generates a public-secret key pair as $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{Setup}}(1^\lambda)$. It computes the obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathcal{O}.\mathsf{Sim}(1^\lambda, 1^s, 1)$, where $s = |\mathsf{P}_{\overline{\mathsf{sk}}}|$. It sends the public key $\mathsf{pk} = (\overline{\mathsf{pk}}, \widetilde{P})$ to $\mathcal{A}$.

We now establish via a sequence of lemmas that the adversary's advantage between Game 1 and 2 is negligible. Let $\mathsf{Adv}^i_{\mathcal{A}} = |\Pr[b' = b] - 1/2|$ denote the advantage of adversary $\mathcal{A}$ in guessing the bit $b$ in Game $i$. We show via a sequence of lemmas that $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ and $\mathsf{Adv}^2_{\mathcal{A}}$ are negligible. Below we discuss our lemmas in detail.

**Lemma 6.1.** If $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ is a secure lockable obfuscator, then for all PPT adversaries $\mathcal{A}$, $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is negligible in the security parameter $\lambda$.

*Proof.* Suppose there exists an adversary $\mathcal{A}$ such that $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is non-negligible. We construct an algorithm $\mathcal{B}$ that can distinguish an obfuscation of program $\mathsf{P}_{\overline{\mathsf{sk}}}$ with a random lock from a simulated obfuscated program, therefore break security of the obfuscation scheme.

$\mathcal{B}$ runs step 1 as in Game 1, except it does not compute obfuscated program $\widetilde{P}$. Concretely, $\mathcal{B}$ chooses a key pair $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{Setup}}(1^\lambda)$. $\mathcal{B}$ sends the program $\mathsf{P}_{\overline{\mathsf{sk}}}$ along with message 1 to the obfuscation challenger, and receives an obfuscated program $\widetilde{P}$. It sets the public key as $\mathsf{pk} = (\overline{\mathsf{pk}}, \widetilde{P})$ and sends $\mathsf{pk}$ to $\mathcal{A}$. It executes steps 2-5 as in Game 1. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '0' to indicate that $\widetilde{P}$ was an obfuscation of $\mathsf{P}_{\overline{\mathsf{sk}}}$; otherwise, it guesses '1' to indicate that it was simulated.

Note that if the obfuscation challenger obfuscated $\mathsf{P}_{\overline{\mathsf{sk}}}$ for some lock $\alpha$, then $\mathcal{B}$ perfectly simulates Game 1 for adversary $\mathcal{A}$. Otherwise it simulates Game 2 for $\mathcal{A}$. As a result, if $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is non-negligible, then $\mathcal{B}$ breaks the obfuscation scheme's security with non-negligible advantage. ∎

**Program** $\mathsf{BP}_{\overline{\mathsf{sk}}}$

**Constants**: Secret key $\overline{\mathsf{sk}}$.
**Inputs:** Ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$.

1. For $i = 1$ to $k$:
   (a) Decrypt $\mathsf{ct}_{i+\ell_1}$ as $\alpha_i = \overline{\mathsf{Dec}}(\mathsf{sk}, \mathsf{ct}_{i+\ell_1})$.
2. If any decryption step fails, output $\perp$. Otherwise, output $\alpha = \alpha_1 \,||\, \cdots \,||\, \alpha_k$.

Figure 4: Bit Encryption 1-Cycle Tester

**Lemma 6.2.** If $\overline{\mathsf{E}} = (\overline{\mathsf{Setup}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ is an IND-CPA secure public key encryption scheme, then for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^2_{\mathcal{A}}$ is negligible in the security parameter $\lambda$.

*Proof.* Suppose there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^2_{\mathcal{A}}$ is non-negligible. We construct an algorithm $\mathcal{B}$ that can break security of the PKE scheme $\overline{\mathsf{E}}$.

$\mathcal{B}$ receives public key $\overline{\mathsf{pk}}$ from $\overline{\mathsf{E}}$ challenger. It runs step 1 as described in Game 2 with the exception that it uses $\overline{\mathsf{pk}}$ generated by $\overline{\mathsf{E}}$ challenger instead of running the setup algorithm. During challenge phase, $\mathcal{B}$ forwards the challenge messages $(m_0, m_1)$ it receives from $\mathcal{A}$ to $\overline{\mathsf{E}}$ challenger as its challenge messages and receives $\mathsf{ct}^*$ as the challenge ciphertext, which it then forwards to $\mathcal{A}$. Finally, $\mathcal{B}$ outputs the same guess as $\mathcal{A}$.

We observe that if $\mathcal{A}$ wins (i.e. $b' = b$), then $\mathcal{B}$ also wins because it exactly simulates the view of Game 2 for $\mathcal{A}$. Therefore if $\mathsf{Adv}^2_{\mathcal{A}}$ is non-negligible, $\mathcal{B}$ must also have non-negligible advantage against $\overline{\mathsf{E}}$ challenger. ■

■

## 6.2 Separating IND-CPA Security from 1-Circular Security for Bit Encryption

In this section, we construct a key cycle tester $\Gamma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Test})$ for 1-bit messages from any public key bit encryption scheme and a lockable obfuscator.

Let $\overline{\mathsf{E}} = (\overline{\mathsf{Setup}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ be a public key bit encryption scheme with secret key space $\{0,1\}^{\ell_1(\lambda)}$, ciphertext space $\{0,1\}^{\ell_2(\lambda)}$ and decryption circuit of depth $d(\lambda)$. Also, let $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ be a lockable obfuscator for 1-bit messages and circuit class $\mathcal{C}_{(\ell_1+k)\cdot\ell_2,k,d}$ (i.e., the class of depth $d(\lambda)$ circuits with $(\ell_1(\lambda)+k(\lambda))\cdot\ell_2(\lambda)$ bit input and $k(\lambda)$ bit output). Below we describe our construction. For notational convenience, let $k = k(\lambda)$, $\ell_1 = \ell_1(\lambda)$, $\ell_2 = \ell_2(\lambda)$ and $n = k(\lambda) + \ell_1(\lambda)$.

- $\mathsf{Setup}(1^\lambda)$ : The setup algorithm runs $\overline{\mathsf{Setup}}$ to obtain a public-secret key pair as $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{Setup}}(1^\lambda)$. It also chooses a string $\alpha$ uniformly at random as $\alpha \leftarrow \{0,1\}^k$.

  Next, consider the program $\mathsf{BP}_{\overline{\mathsf{sk}}}$ described in Figure 4. It obfuscates program $\mathsf{BP}_{\overline{\mathsf{sk}}}$ with message 1 and lock $\alpha$ as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{BP}_{\overline{\mathsf{sk}}}, 1, \alpha)$. Finally, it outputs the key pair as $\mathsf{pk} = (\overline{\mathsf{pk}}, \widetilde{P}), \mathsf{sk} = (\overline{\mathsf{sk}}, \alpha)$.

- $\mathsf{Enc}(\mathsf{pk}, m)$ : Let $\mathsf{pk} = (\overline{\mathsf{pk}}, \widetilde{P})$. The encryption algorithm computes ciphertext as $\mathsf{ct} \leftarrow \overline{\mathsf{Enc}}(\overline{\mathsf{pk}}, m)$.

- $\mathsf{Test}(\mathsf{pk}, \mathbf{ct})$ : Let $\mathsf{pk} = (\overline{\mathsf{pk}}, \widetilde{P})$. The testing algorithm evaluates the obfuscated program $\widetilde{P}$ on input $\mathbf{ct}$ as $b = \mathsf{Eval}(\widetilde{P}, \mathbf{ct})$. If $b = \perp$, it outputs 0. Otherwise, it outputs $b$.

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ : Let $\mathsf{sk} = (\overline{\mathsf{sk}}, \alpha)$. The decryption algorithm outputs $\overline{\mathsf{Dec}}(\overline{\mathsf{sk}}, \mathsf{ct})$.

**Correctness.** For every $\lambda$, public-secret key pair $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{Setup}}(1^\lambda)$ and string $\alpha \leftarrow \{0,1\}^k$, a key cycle consists of a sequence of $n$ ciphertexts $\mathbf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_n)$, where each ciphertext $\mathsf{ct}_i$ is computed as $\mathsf{ct}_i \leftarrow \overline{\mathsf{Enc}}\left(\overline{\mathsf{pk}}, \mathsf{sk}_i\right)$, $\mathsf{sk} = (\mathsf{sk}_t, \alpha)$ and $\mathsf{sk}_i$ is $i^{th}$ bit of $\mathsf{sk}$. Also, the program $\widetilde{P}$ (which is part of public key $\mathsf{pk}$) is computed as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{BP}_{\overline{\mathsf{sk}}}, 1, \alpha)$.

Let $\widetilde{\mathbf{ct}} = (\widetilde{\mathsf{ct}_1}, \ldots, \widetilde{\mathsf{ct}_n})$ be encryptions of zero, where each ciphertext $\widetilde{\mathsf{ct}_i}$ is computed as $\widetilde{\mathsf{ct}_i} \leftarrow \overline{\mathsf{Enc}}(\overline{\mathsf{pk}}, 0)$. For correctness of $\mathsf{Test}$ algorithm, we want to argue that it's advantage in distinguishing a sequence of encryptions of secret key bits from encryptions of zeros is non-negligible.

First, note that $\mathsf{BP}_{\overline{\mathsf{sk}}}(\mathbf{ct}) = \alpha$ and $\mathsf{BP}_{\overline{\mathsf{sk}}}(\widetilde{\mathbf{ct}}) = 0^k$ (by construction). Therefore, by correctness of obfuscation $\mathsf{Eval}(\widetilde{P}, \mathbf{ct}) = 1$. Also, $\mathsf{BP}_{\overline{\mathsf{sk}}}(\widetilde{\mathbf{ct}}) \neq \alpha$ (with all but negligible probability) as $\alpha$ is chosen uniformly at random. This suggests that $\Pr[\mathsf{Eval}(\widetilde{P}, \widetilde{\mathbf{ct}}) = \perp] \geq 1 - \mathrm{negl}(\lambda)$.

Therefore, we can conclude that $\Gamma$ satisfies bit encryption cycle tester correctness condition as it can distinguish a key cycle and all zero encryption with non-negligible probability.

**Security.** We will now show that the scheme described above achieves IND-CPA security as per Definition 2.4. Formally, we prove the following.

**Theorem 6.2.** If $\overline{\mathsf{E}} = (\overline{\mathsf{Setup}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ is an IND-CPA secure public key bit encryption scheme with secret key space $\{0,1\}^{\ell_1(\lambda)}$ and ciphertext space $\{0,1\}^{\ell_2(\lambda)}$ satisfying Definition 2.4, and $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ is a lockable obfuscator for 1-bit messages and circuit class $\mathcal{C}_{(\ell_1+k)\cdot\ell_2,k,d}$ satisfying Definition 3.4, then $\Gamma$ is a secure bit encryption cycle tester scheme satisfying IND-CPA security as per Definition 2.4.

*Proof.* Our proof proceeds via a sequence of hybrid games. Let $\mathcal{A}$ be any PPT adversary that wins the IND-CPA security game against $\Gamma$ with non-negligible advantage. We argue that such an adversary must break security of at least one underlying primitive.

The proof of this theorem is identical to proof of Theorem 6.1. Therefore, we only sketch the sequence of hybrid games.

**Game 1:** This game is the original IND-CPA security game described in Definition 2.4.

1. The challenger generates a public-secret key pair as $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{Setup}}(1^\lambda)$. It uniformly samples $\alpha \leftarrow \{0,1\}^k$, and computes the obfuscation of program $\mathsf{BP}_{\overline{\mathsf{sk}}}$ (described in Figure 4) as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{BP}_{\overline{\mathsf{sk}}}, 1, \alpha)$. It sends the public key $\mathsf{pk} = (\overline{\mathsf{pk}}, \widetilde{P})$ to $\mathcal{A}$.
2. Next, the challenger chooses bit $b \leftarrow \{0,1\}$, computes $\mathsf{ct}^* \leftarrow \overline{\mathsf{Enc}}(\overline{\mathsf{pk}}, b)$, and sends $\mathsf{ct}^*$ to $\mathcal{A}$.
3. $\mathcal{A}$ receives challenge ciphertext $\mathsf{ct}^*$ from the challenger, and outputs its guess $b'$. $\mathcal{A}$ wins if it guesses correctly, that is if $b = b'$.

**Game 2:** This game is same as Game 1, except the challenger does not choose the lock $\alpha$ anymore and it simulates the obfuscated program $\widetilde{P}$ instead of generating it honestly as an obfuscation of program $\mathsf{BP}_{\overline{\mathsf{sk}}}$.

1. The challenger generates a public-secret key pair as $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{Setup}}(1^\lambda)$. It computes the obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathcal{O}.\mathsf{Sim}(1^\lambda, 1^s, 1)$, where $s = |\mathsf{BP}_{\overline{\mathsf{sk}}}|$. It sends the public key $\mathsf{pk} = (\overline{\mathsf{pk}}, \widetilde{P})$ to $\mathcal{A}$.

■

## 6.3 Creating an Unbounded Public Key Cycle Tester

In this section, we construct a cycle tester $\Gamma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Test})$ for unbounded length key cycles from any bootstrappable (leveled) homomorphic encryption scheme and a lockable obfuscator.

Let $\mathsf{LHE} = (\mathsf{LHE.Setup}, \mathsf{LHE.Enc}, \mathsf{LHE.Eval}, \mathsf{LHE.Dec})$ be a bootstrappable homomorphic bit encryption scheme for message space $\{0,1\}^{\ell(\lambda)}$ with decryption circuit of depth $d(\lambda)$, ciphertexts of length $\ell_{\mathsf{ct}}(\lambda)$ and

**Program** $\mathsf{UP}_{\mathsf{lhe.sk}}$

**Constants**: Secret key $\mathsf{lhe.sk}$.
**Inputs:** Ciphertext $\mathsf{ct}$.

1. Decrypt $\mathsf{ct}$ as $(\mathsf{sk}', \alpha) = \mathsf{LHE.Dec}(\mathsf{lhe.sk}, \mathsf{ct})$. Output $\alpha$.

Figure 5: Tester for Unbounded Length Key Cycles

secret keys of length $s(\lambda)$. Also, let $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ be a lockable obfuscator for 1-bit messages and circuit class $\mathcal{C}_{\ell_{\mathsf{ct}},k,d}$ (i.e., the class of depth $d(\lambda)$ circuits with $\ell_{\mathsf{ct}}(\lambda)$ bit input and $k(\lambda)$ bit output). Below we describe our construction. For notational convenience, let $k = k(\lambda)$, $s = s(\lambda)$, $d = d(\lambda)$, $\ell_{\mathsf{ct}} = \ell_{\mathsf{ct}}(\lambda)$ and $\ell = \ell(\lambda)$.

- $\mathsf{Setup}(1^\lambda)$ : The setup algorithm runs $\mathsf{LHE.Setup}$ to obtain a public-secret key pair as $(\mathsf{lhe.pk}, \mathsf{lhe.sk}, \mathsf{lhe.ek}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^d)$. It also chooses a string $\alpha$ uniformly at random as $\alpha \leftarrow \{0,1\}^k$.

  Next, consider the program $\mathsf{UP}_{\mathsf{lhe.sk}}$ described in Figure 5. It obfuscates program $\mathsf{UP}_{\mathsf{lhe.sk}}$ with message 1 and lock $\alpha$ as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{UP}_{\mathsf{lhe.sk}}, 1, \alpha)$. Finally, it outputs the key pair as $\mathsf{pk} = (\mathsf{lhe.pk}, \mathsf{lhe.ek}, \widetilde{P}), \mathsf{sk} = (\mathsf{lhe.sk}, \alpha)$.

- $\mathsf{Enc}(\mathsf{pk}, m)$ : Let $\mathsf{pk} = (\mathsf{lhe.pk}, \mathsf{lhe.ek}, \widetilde{P})$. The encryption algorithm computes ciphertext as $\mathsf{ct} \leftarrow \mathsf{LHE.Enc}(\mathsf{lhe.pk}, m)$.

- $\mathsf{Test}(1^n, \mathbf{pk}, \mathbf{ct})$ : Let $\mathbf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$, $\mathbf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_n)$ and $\mathsf{pk}_i = (\mathsf{lhe.pk}_i, \mathsf{lhe.ek}_i, \widetilde{P}_i)$ (for all $i \leq n$). Also, let circuit $C_{\mathsf{ct}}$ be a circuit which takes as input $s + k$ bits and decrypts ciphertext $\mathsf{ct}$ using the first $s$ bits of the input, i.e. $C_{\mathsf{ct}}(x \,||\, y) = \mathsf{LHE.Dec}(x, \mathsf{ct})$ where $x \in \{0,1\}^s, y \in \{0,1\}^k$. The tester algorithm runs as follows.

  First, it sets $\mathsf{ct}'_n = \mathsf{ct}_n$. For $i = n-1$ to $i = 1$, it computes ciphertexts $\mathsf{ct}'_i$ as $\mathsf{ct}'_i = \mathsf{LHE.Eval}(\mathsf{lhe.ek}_i, C_{\mathsf{ct}'_{i+1}}, \mathsf{ct}_i)$. Next, it evaluates the obfuscated program $\widetilde{P}_1$ on input $\mathsf{ct}'_1$ as $b = \mathsf{Eval}(\widetilde{P}_1, \mathsf{ct}'_1)$. Finally, it outputs $b$.

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ : Let $\mathsf{sk} = (\mathsf{lhe.sk}, \alpha)$. The decryption algorithm outputs $\mathsf{LHE.Dec}(\mathsf{lhe.sk}, \mathsf{ct})$.

**Correctness.** For all $\lambda, n \in \mathbb{N}$, LHE key pairs $(\mathsf{lhe.pk}_i, \mathsf{lhe.sk}_i, \mathsf{lhe.ek}_i) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^d)$ and strings $\alpha_i \leftarrow \{0,1\}^k$ (for $i \leq n$), a key cycle of length $n$ consists of a sequence of $n$ ciphertexts $\mathbf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_n)$, where each ciphertext $\mathsf{ct}_i$ is computed as $\mathsf{ct}_i \leftarrow \overline{\mathsf{Enc}}\left(\mathsf{lhe.pk}_i, (\mathsf{lhe.sk}_{(i \bmod n)+1}, \alpha_{(i \bmod n)+1})\right)$. Also, the program $\widetilde{P}_1$ (which is part of public key $\mathsf{pk}_1$) is computed as $\widetilde{P}_1 \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{UP}_{\mathsf{lhe.sk}_1}, 1, \alpha_1)$.

Let $\widetilde{\mathbf{ct}} = (\widetilde{\mathsf{ct}_1}, \ldots, \widetilde{\mathsf{ct}_n})$ be encryption of zero strings, where each ciphertext $\widetilde{\mathsf{ct}_i}$ is computed as $\widetilde{\mathsf{ct}_i} \leftarrow \overline{\mathsf{Enc}}(\overline{\mathsf{pk}}_i, 0^{s+k})$. For correctness of $\mathsf{Test}$ algorithm, we want to argue that it's advantage in distinguishing a sequence of encryptions of secret keys from encryptions of zeros is non-negligible.

First, note that $\mathsf{LHE.Dec}(\mathsf{lhe.sk}_n, \mathsf{ct}_n) = (\mathsf{sk}_1, \alpha_1)$. Therefore, $\mathsf{ct}'_{n-1} = \mathsf{LHE.Eval}(\mathsf{lhe.ek}_{n-1}, C_{\mathsf{ct}'_n}, \mathsf{ct}_{n-1})$ is an encryption of $(\mathsf{sk}_1, \alpha_1)$ under key $\mathsf{lhe.pk}_{n-1}$, where $\mathsf{ct}'_n = \mathsf{ct}_n$. Therefore, the top-down iterative LHE evaluation during testing would finally result in ciphertext $\mathsf{ct}'_1$ that will be an encryption of $(\mathsf{sk}_1, \alpha_1)$ under key $\mathsf{lhe.pk}_1$. We could prove this by an inductive argument over cycle length $n$. The base case $n = 1$ follows directly from the construction. The inductive hypothesis would be that the test algorithm successfully reduces a length $k$ key cycle to a length 1 key cycle. The induction step can be proven by first reducing a length $k$ key cycle to a length $k-1$ key cycle (which follows from the correctness of the LHE scheme) and then applying the hypothesis. Finally, the test algorithm evaluates the obfuscated program $\widetilde{P}_1$ on $\mathsf{ct}'_1$ which is encryption of $(\mathsf{sk}_1, \alpha_1)$ under key $\mathsf{lhe.pk}_1$. Therefore, by correctness of obfuscation scheme $\mathsf{Eval}(\widetilde{P}_1, \mathsf{ct}'_1) = 1$. In other words, we can claim that the test algorithm always outputs 1 in this case (i.e., with probability 1).

Similarly, $\mathsf{UP}_{\mathsf{lhe.sk}_1}(\widetilde{\mathsf{ct}}'_1) \neq \alpha_1$ (with all but negligible probability) as $\alpha_1$ is chosen uniformly at random and neither $\mathsf{UP}_{\mathsf{lhe.sk}_1}$ nor any of the ciphertexts $\mathsf{ct}_i$ depends on the string $\alpha_1$. This suggests that $\Pr[\mathsf{Eval}(\widetilde{P}_1, \widetilde{\mathsf{ct}}'_1) = \perp] \geq 1 - \mathrm{negl}(\lambda)$.

Therefore, we can conclude that $\Gamma$ satisfies cycle tester correctness condition as it can distinguish key cycles and all zero encryption with non-negligible probability.

The IND-CPA proof of this scheme is identical to the proof of Theorem 6.1.

# 7 Uninstantiability of the Fujisaki-Okamoto and Related Transformations

We now move to our second set of separation results where we use lockable obfuscation to show random oracle uninstantiability [CGH98] for a certain family of transformations. These include: the Bellare et al. [BHSV98] transformation from an IND-CPA scheme to an injective trapdoor function, the well known Fujisaki-Okamoto [FO99a, FO99b] transformations from IND-CPA and the deterministic encryption construction of Bellare, Boldyreva and O'Neill [BBO07].

All of the constructions follow a similar paradigm where they encrypt a string $x$ under random coins determined from $H(x)$. (How the string $x$ is construed varies somewhat between the schemes.) The works above show that if $H$ is presented as an oracle access to a random function, the transformation results in a secure scheme under the relevant definition.

An important question is what can be said about the security of these transformations when $H$ is instantiated by a concrete hash function. Assuming indistinguishability obfuscation Brzuska, Farshim and Mittelbach [BFM15] provide negative results on the instantiability of these transformations. In particular assuming iO for circuits they provide encryption schemes that are IND-CPA secure, but where the transformation is insecure when instantiated with any hash function up to an apriori bounded size. Furthermore, if one assumes iO for Turing Machines with unbouded input, the result holds for hash functions of any size.[15] The main idea of BFM is to create an encryption scheme that includes an obfuscated program which can be used to break security when the description of $H$ is known.

Here we show that such results can be achieved from lockable obfuscation and thus the Learning with Errors assumption. More specifically, the uninstantiability results for apriori size bounded hash functions follows from lockable obfuscation and the unrestricted results follow from lockable obfuacation combined with fully homomorphic encryption. The main insight into the result is simply that the program breaking in BFM falls within a pattern that matches lockable obfuscation. Once this connection is made the technical argument follows readily. In addition, we again see that for iO programs that match the pattern of lockable obfuscation, fully homomorphic encryption can serve to give the "Turing Machine version" of such programs.

Below we give a proof of uninstantiability for an IND-CPA encryption scheme for the second Fujisaki-Okamoto [FO99b] transformation. We do so in the unrestricted size case using lockable obfuscation along with fully homomorphic encryption. We believe uninstantiability results for the other schemes mentioned above follow in a very similar manner.

## 7.1 The Fujisaki-Okamoto Transformation

The Fujisaki-Okamoto (FO) transformation is a technique to convert an IND-CPA secure public key encryption scheme into an IND-CCA secure public key encryption scheme.[16] The transformation relies on a public key encryption scheme, a symmetric encryption scheme and two hash functions (which are modeled as two independent random oracles).

Let $\mathsf{PKE}$ = ($\mathsf{PKE.Setup}$, $\mathsf{PKE.Enc}$, $\mathsf{PKE.Dec}$) be a public key encryption scheme, $\mathsf{SKE}$ = ($\mathsf{SKE.Setup}$, $\mathsf{SKE.Enc}$, $\mathsf{SKE.Dec}$) be a (deterministic) symmetric encryption scheme, and $H_1, H_2$ be two hash functions. As per FO transformation, an encryption of message $m$ under randomness $r$ is computed as

$$\mathsf{PKE.Enc}(\mathsf{pke.pk}, r; H_1(m \,||\, r)), \mathsf{SKE.Enc}(\mathsf{ske.sk} = H_2(r), m).$$

[15] We note that constructing Turing Machine iO for *unbounded* length inputs from standard iO remains an open problem.

[16] The FO transformation only requires the starting PKE scheme to satisfy one-way security, which is a slightly weaker notion than IND-CPA.

More formally, the FO transformed scheme $\mathsf{FO}^{\mathcal{H},\widetilde{\mathcal{H}}}_{\mathsf{PKE},\mathsf{SKE}} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ is described as follows.

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$. It chooses two hash functions $H_1, H_2$ as $H_1 \leftarrow \mathcal{H}_\lambda, H_2 \leftarrow \widetilde{\mathcal{H}}_\lambda$. It also samples a public-secret key pair $(\mathsf{pke.pk}, \mathsf{pke.sk})$ for PKE scheme as $(\mathsf{pke.pk}, \mathsf{pke.sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$. It outputs the public-secret keys as $\mathsf{pk} = (\mathsf{pke.pk}, H_1, H_2)$ and $\mathsf{sk} = (\mathsf{pke.pk}, H_1, H_2, \mathsf{pke.sk})$.
- $\mathsf{Enc}(\mathsf{pk}, m; r) \to \mathsf{ct}$. Let $\mathsf{pk} = (\mathsf{pke.pk}, H_1, H_2)$. It computes ciphertext $\mathsf{ct}_1, \mathsf{ct}_2$ as $\mathsf{ct}_1 = \mathsf{PKE.Enc}(\mathsf{pke.pk}, r; H_1(m \,||\, r))$ and $\mathsf{ct}_2 = \mathsf{SKE.Enc}(H_2(r), m)$. It outputs the ciphertext as $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m$ or $\perp$ . Let $\mathsf{sk} = (\mathsf{pke.pk}, H_1, H_2, \mathsf{pke.sk})$ and $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$. It decrypts $\mathsf{ct}_1$ as $r = \mathsf{PKE.Dec}(\mathsf{pke.sk}, \mathsf{ct}_1)$. Next, it decrypts $\mathsf{ct}_2$ as $m = \mathsf{SKE.Dec}(H_2(r), \mathsf{ct}_2)$. Finally, it checks if $\mathsf{ct}_1 = \mathsf{PKE.Enc}(\mathsf{pke.pk}, r; H_1(m \,||\, r))$. If the check succeeds it outputs $m$, otherwise it outputs $\perp$.

In this work, we show that FO transform is uninstantiable in the standard model. Concretely, we prove the following.

**Theorem 7.1.** If $\mathsf{FHE} = (\mathsf{FHE.Setup}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ is a fully homomorphic encryption scheme satisfying Definition 2.4, and $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ is a lockable obfuscator for circuit class $\{\mathcal{C}_{\ell.\widetilde{\ell},\ell,d}\}_\lambda$ satisfying Definition 3.4, then there exists a public key encryption scheme $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ such that for every hash function family $\mathcal{H} = \{\mathcal{H}_\lambda\}_\lambda$ and $\widetilde{\mathcal{H}} = \left\{\widetilde{\mathcal{H}}_\lambda\right\}_\lambda$, symmetric key encryption scheme $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$, the corresponding FO transformed scheme $\mathsf{FO}^{\mathcal{H},\widetilde{\mathcal{H}}}_{\mathsf{PKE},\mathsf{SKE}}$ is not secure.[17]

We prove the above theorem by constructing such a public key encryption scheme $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$. Consider a public key encryption scheme $\overline{\mathsf{PKE}} = (\overline{\mathsf{PKE.Setup}}, \overline{\mathsf{PKE.Enc}}, \overline{\mathsf{PKE.Dec}})$ for $\ell$-bit messages, a fully homomorphic encryption scheme $\mathsf{FHE} = (\mathsf{FHE.Setup}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ for 1-bit messages with decryption circuit of depth $d(\lambda)$ and ciphertext space $\{0,1\}^{\widetilde{\ell}(\lambda)}$, and a lockable obfuscator scheme for $\ell$-bit messages. Below we describe our construction. For ease of exposition, assume that the FHE setup, FHE encryption, PKE encryption and lockable obfuscator also take $\ell$ bits of randomness as inputs.

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$. It generates a PKE key pair as $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{PKE.Setup}}(1^\lambda)$, and sets public-secret key pair as $(\mathsf{pk}, \mathsf{sk}) = (\overline{\mathsf{pk}}, \overline{\mathsf{sk}})$.
- $\mathsf{Enc}(\mathsf{pk}, m; r) \to \mathsf{ct}$. Let $\mathsf{pk} = \overline{\mathsf{pk}}$ and $r = r_0 \,||\, r_1 \,||\, r_2 \,||\, r_3 \,||\, r_4$, where each $r_i$ is $\ell$ bits long. It samples an FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek}) = \mathsf{FHE.Setup}(1^\lambda; r_0)$ and computes ciphertext $\mathsf{ct}_1$ as $\mathsf{ct}_1 = \overline{\mathsf{PKE.Enc}}(\overline{\mathsf{pk}}, m; r_1)$. It encrypts message $m$ under FHE public key as $\mathbf{ct}_2 = \mathsf{FHE.Enc}(\mathsf{fhe.pk}, m; r_2)$.[18]

  It also obfuscates program $\mathsf{P}_{\mathsf{fhe.sk}}$ with message $m$, lock $r_4$ and randomness $r_3$ as $\widetilde{P} = \mathsf{Obf}(1^\lambda, \mathsf{P}_{\mathsf{fhe.sk}}, m, r_4; r_3)$. Finally, it outputs the ciphertext as $\mathsf{ct} = (\mathsf{fhe.ek}, \mathsf{ct}_1, \mathbf{ct}_2, \widetilde{P})$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m$. Let $\mathsf{ct} = (\mathsf{fhe.ek}, \mathsf{ct}_1, \mathbf{ct}_2, \widetilde{P})$. It decrypts $\mathsf{ct}_1$ as $m = \overline{\mathsf{PKE.Dec}}(\mathsf{sk}, \mathsf{ct}_1)$ and outputs $m$.

First, we prove that the above scheme is IND-CPA secure.

**Theorem 7.2.** If $\mathsf{FHE} = (\mathsf{FHE.Setup}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ is a fully homomorphic encryption scheme satisfying Definition 2.4, $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ is a lockable obfuscator for circuit class $\{\mathcal{C}_{\ell.\widetilde{\ell},\ell,d}\}_\lambda$ satisfying Definition 3.4, and $\overline{\mathsf{PKE}} = (\overline{\mathsf{PKE.Setup}}, \overline{\mathsf{PKE.Enc}}, \overline{\mathsf{PKE.Dec}})$ is a public key encryption scheme for $\ell$-bit messages satisfying Definition 2.4, then the scheme $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ is IND-CPA secure.

*Proof.* Our proof proceeds via a sequence of hybrid games. Let $\mathcal{A}$ be any PPT adversary that wins the IND-CPA security game against $\mathsf{PKE}$ with non-negligible advantage. We argue that such an adversary must break security of at least one underlying primitive.

We will first define the sequence of hybrid games, and then show that they are computationally indistinguishable.

[17] The FO transformed scheme $\mathsf{FO}^{\mathcal{H},\widetilde{\mathcal{H}}}_{\mathsf{PKE},\mathsf{SKE}}$ does not even achieve IND-CPA security.

[18] Note that FHE scheme is a bit encryption scheme, therefore it encrypts one bit at a time. However, we overload the notation and use $\mathsf{FHE.Enc}$ to encrypt multi-bit messages as well. So, $\mathbf{ct}_2$ is a sequence of $\ell$ FHE ciphertexts.

**Program** $\mathsf{P}_{\mathsf{fhe.sk}}$

**Constants**: Secret key $\mathsf{fhe.sk}$.
**Inputs:** Ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$.

1. Decrypt $\mathsf{ct}_i$ as $\alpha_i = \mathsf{FHE.Dec}(\mathsf{fhe.sk}, \mathsf{ct}_i)$.
2. If any decryption step fails, output $\perp$. Otherwise, output $\alpha_1 || \cdots || \alpha_\ell$.

Figure 6: FO

**Game 1:** This game is the original IND-CPA security game described in Definition 2.4.

1. The challenger generates a public-secret key pair as $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{PKE}}.\overline{\mathsf{Setup}}(1^\lambda)$. It sends the public key $\mathsf{pk} = \overline{\mathsf{pk}}$ to $\mathcal{A}$.
2. $\mathcal{A}$ receives $\mathsf{pk}$ from the challenger, and computes messages $m_0, m_1$. It sends $(m_0, m_1)$ as its challenge messages to the challenger.
3. The challenger chooses bit $b \leftarrow \{0,1\}$. It samples an FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$ and computes ciphertexts $\mathsf{ct}_1, \mathbf{ct}_2$ as $\mathsf{ct}_1 \leftarrow \overline{\mathsf{PKE}}.\overline{\mathsf{Enc}}(\overline{\mathsf{pk}}, m_b), \mathbf{ct}_2 \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, m_b)$. It chooses a random $\ell$ bit string $r$ and obfuscates program $\mathsf{P}_{\mathsf{fhe.sk}}$ with message $m$ and lock $r$ as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{P}_{\mathsf{fhe.sk}}, m_b, r)$. Finally, it outputs the ciphertext as $\mathsf{ct} = (\mathsf{fhe.ek}, \mathsf{ct}_1, \mathbf{ct}_2, \widetilde{P})$. Finally, it sends $\mathsf{ct}$ to $\mathcal{A}$.
4. $\mathcal{A}$ receives challenge ciphertext $\mathsf{ct}$ from the challenger, and outputs its guess $b'$.
5. $\mathcal{A}$ wins if it guesses correctly, that is if $b = b'$.

**Game 2:** This game is same as Game 1, except the challenger does not choose the lock $r$ anymore and it simulates the obfuscated program $\widetilde{P}$ instead of generating it honestly.

1. The challenger chooses bit $b \leftarrow \{0,1\}$. It samples an FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$ and computes ciphertexts $\mathsf{ct}_1, \mathbf{ct}_2$ as $\mathsf{ct}_1 \leftarrow \overline{\mathsf{PKE}}.\overline{\mathsf{Enc}}(\overline{\mathsf{pk}}, m_b), \mathbf{ct}_2 \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, m_b)$. It computes the obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathcal{O}.\mathsf{Sim}(1^\lambda, 1^s, 1^\ell)$, where $s = |\mathsf{P}_{\mathsf{fhe.sk}}|$. Finally, it outputs the ciphertext as $\mathsf{ct} = (\mathsf{fhe.ek}, \mathsf{ct}_1, \mathbf{ct}_2, \widetilde{P})$. Finally, it sends $\mathsf{ct}$ to $\mathcal{A}$.

**Game 3:** This game is same as Game 2, except the challenger computes $\mathbf{ct}_2$ as encryption of all zeros string.

1. The challenger chooses bit $b \leftarrow \{0,1\}$. It samples an FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$ and computes ciphertexts $\mathsf{ct}_1, \mathbf{ct}_2$ as $\mathsf{ct}_1 \leftarrow \overline{\mathsf{PKE}}.\overline{\mathsf{Enc}}(\overline{\mathsf{pk}}, m_b), \mathbf{ct}_2 \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, 0^\ell)$. It computes the obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathcal{O}.\mathsf{Sim}(1^\lambda, 1^s, 1^\ell)$, where $s = |\mathsf{P}_{\mathsf{fhe.sk}}|$. Finally, it outputs the ciphertext as $\mathsf{ct} = (\mathsf{fhe.ek}, \mathsf{ct}_1, \mathbf{ct}_2, \widetilde{P})$. Finally, it sends $\mathsf{ct}$ to $\mathcal{A}$.

We now establish via a sequence of claims that the adversary's advantage between any two consecutive games is negligible. Let $\mathsf{Adv}^i_{\mathcal{A}} = |\Pr[b' = b] - 1/2|$ denote the advantage of adversary $\mathcal{A}$ in guessing the bit $b$ in Game $i$. We show via a sequence of lemmas that $|\mathsf{Adv}^i_{\mathcal{A}} - \mathsf{Adv}^{i+1}_{\mathcal{A}}|$ (for $i = 1, 2$) and $\mathsf{Adv}^3_{\mathcal{A}}$ are negligible. Below we discuss our lemmas in detail.

**Lemma 7.1.** If $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ is a secure lockable obfuscator, then for all PPT adversaries $\mathcal{A}$, $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is negligible in the security parameter $\lambda$.

*Proof.* Suppose there exists an adversary $\mathcal{A}$ such that $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is non-negligible. We construct an algorithm $\mathcal{B}$ that can distinguish an obfuscation of program $\mathsf{P}_{\mathsf{fhe.sk}}$ with a random lock from a simulated obfuscated program, therefore break security of the obfuscation scheme.

$\mathcal{B}$ runs step 1 as in Game 1, i.e. it samples a PKE public-secret key pair $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}})$ and sends $\overline{\mathsf{pk}}$ to $\mathcal{A}$. Next, it receives two challenge messages $(m_0, m_1)$ from $\mathcal{A}$. $\mathcal{B}$ chooses a FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$ and random bit $b$. $\mathcal{B}$ sends the program $\mathsf{P}_{\mathsf{fhe.sk}}$ along with message $m_b$ to the obfuscation challenger, and receives an obfuscated program $\widetilde{P}$. It encrypts message $m_b$ as $\mathsf{ct}_1 \leftarrow \overline{\mathsf{PKE}}.\overline{\mathsf{Enc}}(\overline{\mathsf{pk}}, m_b), \mathbf{ct}_2 \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, m_b)$. Finally, it sends the challenge ciphertext $\mathsf{ct} = (\mathsf{fhe.ek}, \mathsf{ct}_1, \mathbf{ct}_2, \widetilde{P})$ to $\mathcal{A}$. Next, $\mathcal{B}$ receives $\mathcal{A}$'s guess $b'$. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '0' to indicate that $\widetilde{P}$ was an obfuscation of $\mathsf{P}_{\mathsf{fhe.sk}}$; otherwise, it guesses '1' to indicate that it was simulated.

Note that if the obfuscation challenger obfuscated $\mathsf{P}_{\mathsf{fhe.sk}}$ for some lock $r$, then $\mathcal{B}$ perfectly simulates Game 1 for adversary $\mathcal{A}$. Otherwise it simulates Game 2 for $\mathcal{A}$. As a result, if $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is non-negligible, then $\mathcal{B}$ breaks the obfuscation scheme's security with non-negligible advantage. ■

**Lemma 7.2.** If $\mathsf{FHE} = (\mathsf{FHE.Setup}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ is a fully homomorphic encryption scheme, then for all PPT adversaries $\mathcal{A}$, $|\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}}|$ is negligible in the security parameter $\lambda$.

*Proof.* Suppose there exists an adversary $\mathcal{A}$ such that $|\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}}|$ is non-negligible. We construct an algorithm $\mathcal{B}$ that can distinguish encryption of message $m_b$ and an all zeros string, therefore break security of the FHE scheme.

$\mathcal{B}$ receives an FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.ek})$ from the challenger. $\mathcal{B}$ runs step 1 as in Game 1, i.e. it samples a PKE public-secret key pair $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}})$ and sends $\overline{\mathsf{pk}}$ to $\mathcal{A}$. Next, it receives two challenge messages $(m_0, m_1)$ from $\mathcal{A}$. $\mathcal{B}$ computes the obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathcal{O}.\mathsf{Sim}(1^\lambda, 1^s, 1^\ell)$, where $s = |\mathsf{P}_{\mathsf{fhe.sk}}|$. It chooses a random bit $b$ and encrypts message $m_b$ as $\mathsf{ct}_1 \leftarrow \overline{\mathsf{PKE}}.\overline{\mathsf{Enc}}(\overline{\mathsf{pk}}, m_b)$. It sends $(m_b, 0^\ell)$ to the FHE challenger as its challenge messages. $\mathcal{B}$ receives $\mathbf{ct}^*$ from the challenger and sets $\mathbf{ct}_2 = \mathbf{ct}^*$. Finally, it sends the challenge ciphertext $\mathsf{ct} = (\mathsf{fhe.ek}, \mathsf{ct}_1, \mathbf{ct}_2, \widetilde{P})$ to $\mathcal{A}$. Next, $\mathcal{B}$ receives $\mathcal{A}$'s guess $b'$. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ sends 0 as its guess (i.e., $m_b$ was encrypted), otherwise it sends 1 (i.e., $0^\ell$ was encrypted) as its guess to the FHE challenger.

Note that if the FHE challenger encrypted $m_b$, then $\mathcal{B}$ perfectly simulates Game 2 for adversary $\mathcal{A}$. Otherwise it simulates Game 3 for $\mathcal{A}$. Therefore, if $|\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}}|$ is non-negligible, then $\mathcal{B}$ breaks the FHE scheme's security with non-negligible advantage. ■

**Lemma 7.3.** If $\overline{\mathsf{PKE}} = (\overline{\mathsf{PKE}}.\overline{\mathsf{Setup}}, \overline{\mathsf{PKE}}.\overline{\mathsf{Enc}}, \overline{\mathsf{PKE}}.\overline{\mathsf{Dec}})$ is an IND-CPA secure public key encryption scheme, then for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^3_{\mathcal{A}}$ is negligible in the security parameter $\lambda$.

*Proof.* Suppose there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^3_{\mathcal{A}}$ is non-negligible. We construct an algorithm $\mathcal{B}$ that can break security of the PKE scheme $\overline{\mathsf{PKE}}$.

$\mathcal{B}$ receives public key $\overline{\mathsf{pk}}$ from the PKE challenger. It sends $\overline{\mathsf{pk}}$ to $\mathcal{A}$. Next, it receives two challenge messages $(m_0, m_1)$ from $\mathcal{A}$. $\mathcal{B}$ chooses a FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$. It computes the obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathcal{O}.\mathsf{Sim}(1^\lambda, 1^s, 1^\ell)$, where $s = |\mathsf{P}_{\mathsf{fhe.sk}}|$. It also computes ciphertext $\mathbf{ct}_2$ as $\mathbf{ct}_2 \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, 0^\ell)$. It sends $(m_0, m_1)$ to the PKE challenger as its challenge messages. $\mathcal{B}$ receives $\mathsf{ct}^*$ from the challenger and sets $\mathsf{ct}_1 = \mathsf{ct}^*$. Finally, it sends the challenge ciphertext $\mathsf{ct} = (\mathsf{fhe.ek}, \mathsf{ct}_1, \mathbf{ct}_2, \widetilde{P})$ to $\mathcal{A}$. Next, $\mathcal{B}$ receives $\mathcal{A}$'s guess $b'$. Finally, $\mathcal{B}$ outputs the same guess as $\mathcal{A}$.

We observe that if $\mathcal{A}$ wins (i.e. $b' = b$), then $\mathcal{B}$ also wins because it exactly simulates the view of Game 3 for $\mathcal{A}$. Therefore if $\mathsf{Adv}^3_{\mathcal{A}}$ is non-negligible, $\mathcal{B}$ must also have non-negligible advantage against $\overline{\mathsf{PKE}}$ challenger. ■

This concludes the proof of IND-CPA security of $\mathsf{PKE}$.

■

**Uninstantiability of FO Transformation.** Now, we show that for every hash function family $\mathcal{H} = \{\mathcal{H}_\lambda\}_\lambda$ and $\widetilde{\mathcal{H}} = \left\{\widetilde{\mathcal{H}}_\lambda\right\}_\lambda$, symmetric key encryption scheme $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$, the FO transformed scheme $\mathsf{FO}^{\mathcal{H},\widetilde{\mathcal{H}}}_{\mathsf{PKE},\mathsf{SKE}}$ is not IND-CPA secure. Below we describe an attacker $\mathcal{A}$.

1. The challenger chooses two hash functions $H_1, H_2$ as $H_1 \leftarrow \mathcal{H}_\lambda, H_2 \leftarrow \widetilde{\mathcal{H}}_\lambda$. It samples a public-secret key pair $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}})$ for PKE scheme $\overline{\mathsf{PKE}}$ as $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{PKE}}.\overline{\mathsf{Setup}}(1^\lambda)$. It sends the public key $\overline{\mathsf{pk}}$ and hash function descriptions $H_1, H_2$ to adversary $\mathcal{A}$.

2. $\mathcal{A}$ chooses two $\ell$-bit messages $m_0, m_1 (\neq m_0)$ uniformly at random and sends them as its challenge messages.

3. The challenger chooses a random bit $b$ and an $\ell$-bit string $s \leftarrow \{0,1\}^\ell$. It computes $r = H_1(m_b \,||\, s)$. Let $r = r_0 \,||\, r_1 \,||\, r_2 \,||\, r_3 \,||\, r_4$, whereeach $r_i$ is $\ell$ bits long.

   It samples an FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek}) = \mathsf{FHE.Setup}(1^\lambda; r_0)$ and computes ciphertext $\mathsf{ct}_1$ as $\mathsf{ct}_1 = \overline{\mathsf{PKE}}.\overline{\mathsf{Enc}}(\overline{\mathsf{pk}}, s; r_1)$. It also encrypts $s$ under FHE public key as $\mathbf{ct}_2 = \mathsf{FHE.Enc}(\mathsf{fhe.pk}, s; r_2)$. It obfuscates program $\mathsf{P}_{\mathsf{fhe.sk}}$ with message $s$, lock $r_4$ and randomness $r_3$ as $\widetilde{P} = \mathsf{Obf}(1^\lambda, \mathsf{P}_{\mathsf{fhe.sk}}, s, r_4; r_3)$. This corresponds to the PKE part of the ciphertext, i.e. $\mathsf{ct}_{\mathsf{PKE}} = (\mathsf{fhe.ek}, \mathsf{ct}_1, \mathbf{ct}_2, \widetilde{P})$.

   Next, it encrypts message $m_b$ using SKE scheme as $\mathsf{ct}_{\mathsf{SKE}} = \mathsf{SKE.Enc}(H_2(s), m_b)$. Finally, it sends $\mathsf{ct} = (\mathsf{ct}_{\mathsf{PKE}}, \mathsf{ct}_{\mathsf{SKE}})$ as the challenge ciphertext to $\mathcal{A}$.

4. $\mathcal{A}$ receives the ciphertext $\mathsf{ct} = ((\mathsf{fhe.ek}, \mathsf{ct}_1, \mathbf{ct}_2, \widetilde{P}), \mathsf{ct}_{\mathsf{SKE}})$ from the challenger. It first homomorphically evaluates the hash function $H_2$ on $\mathbf{ct}_2$ and then decrypts the resulting string to decrypt ciphertext $\mathsf{ct}_{\mathsf{SKE}}$. Concretely, it computes $\mathbf{ct}' = \mathsf{FHE.Eval}(\mathsf{fhe.ek}, \mathsf{SKE.Dec}(H_2(\cdot), \mathsf{ct}_{\mathsf{SKE}}), \mathbf{ct}_2)$. Next, it homomorphically evaluates hash function $H_1$ on $\mathbf{ct}', \mathbf{ct}_2$ as $\widetilde{\mathbf{ct}} = \mathsf{FHE.Eval}(\mathsf{fhe.ek}, H_1(\cdot), (\mathbf{ct}', \mathbf{ct}_2))$. Let $\widetilde{\mathbf{ct}} = \widetilde{\mathsf{ct}}_1, \ldots, \widetilde{\mathsf{ct}}_{5\ell}$.

   It evaluates program $\widetilde{P}$ on ciphertexts $\widetilde{\mathsf{ct}}_{4\ell+1}, \ldots, \widetilde{\mathsf{ct}}_{5\ell}$ as $t = \mathsf{Eval}(\widetilde{P}, (\widetilde{\mathsf{ct}}_{4\ell+1}, \ldots, \widetilde{\mathsf{ct}}_{5\ell}))$. It decrypts ciphertext $\mathsf{ct}_{\mathsf{SKE}}$ as $m = \mathsf{SKE.Dec}(H_2(t), \mathsf{ct}_{\mathsf{SKE}})$. If $m = m_0$ or $m = m_1$, then it guesses accordingly. Otherwise, it guesses randomly.

We claim that $\mathcal{A}$ always guesses $b$ correctly. In other words, $\mathcal{A}$ wins with non-negligible advantage.
First, note that (by correctness of FHE evaluation) ciphertext $\mathbf{ct}'$ encrypts challenge message $m_b$ as $H_2(s)$ is the SKE secret key used for encrypting $m_b$. Similarly, ciphertext $\widetilde{\mathbf{ct}}$ is an encryption of string $r = H_1(m_b \,||\, s)$. Therefore, ciphertexts $\widetilde{\mathsf{ct}}_{4\ell+1}, \ldots, \widetilde{\mathsf{ct}}_{5\ell}$ are encryptions of each bit of $r_4$. Since $\mathsf{P}_{\mathsf{fhe.sk}}(\widetilde{\mathsf{ct}}_{4\ell+1}, \ldots, \widetilde{\mathsf{ct}}_{5\ell}) = r_4$. thus, by correctness of obfuscation, we know that $\mathsf{Eval}(\widetilde{P}, (\widetilde{\mathsf{ct}}_{4\ell+1}, \ldots, \widetilde{\mathsf{ct}}_{5\ell})) = s$. In other words, $t = s$. Finally, by correctness of SKE scheme, we can conclude that $\mathsf{SKE.Dec}(H_2(t), \mathsf{ct}_{\mathsf{SKE}}) = m_b$. Therefore, $\mathcal{A}$ always guesses correctly.

We would like to point out that $\mathcal{A}$ learns the challenge message completely, therefore we could also prove that scheme $\mathsf{FO}^{\mathcal{H},\widetilde{\mathcal{H}}}_{\mathsf{PKE},\mathsf{SKE}}$ is not even one-way secure.

# 8 Indistinguishability Obfuscation for Rejecting Programs

We now consider a new notition of obfucation that we call indistinguishability obfuscation for rejecting programs and show how to construct it from lockable obfuscation and witness encryption for circuit satisfiability.

Obfuscators that meet this notion will be defined over boolean circuits. Like indistinguishability obfuscation our obfuscator will take in any (not necessarily rejecting) boolean circuit $C$ in a class and output an obfuscated program that is functionally equivalent to $C$. However, the security guarantees given by such an obfuscator are limited to "rejecting" programs. Informally, they state that no PPT adversary can distinguish between circuits $C_0$ and $C_1$ as long as for all inputs $x$ $C_0(x) = C_1(x) = 0$. In contrast, standard indistinguishability obfuscation security allows $C_0, C_1$ to have arbitrary (both 0 and 1) outputs as long as they are functionally equivalent.

Our construction is simple and follows along the same conceptual lines as our techniques for attribute hiding from Section 5. Recall, that in a witness encryption scheme for circuit satisfiability the encryption operation will take as input a circuit description $C$ as well as a message $\mathsf{msg}$. The message will be computationally hidden as long as there is no satisfying assignment to $C$. I.e. for all $x$ we have $C(x) = 0$. However, there is nothing in the definition of witness encryption that guarantees the hiding of the description of the circuit $C$ itself.

To obfuscate a circuit $C$ one first chooses a random lock value $\alpha$ and creates a witness encryption of the message $\alpha$ under circuit $C$ to get a (sub) ciphertext $\mathsf{ct}$. Then we can create a lockable obfuscation of the following program. The program on input $x$ will first check if $C(x) = 0$, and if so it outputs $0^{\ell_{\text{out}}}$. Otherwise, it runs the witness encryption decryption algorithm with $x$ as the witness that $C$ is satisfiable. It is easy to veryify correctness. On any input $x$ where $C(x) = 0$ the lockable program will reject and we can output 0. On the other hand if $C(x) = 1$ the internal decryption will result in the lock value $\alpha$ and a message will be output. The security argument follows readily and is given below.

While the idea of indistinguishability for rejecting programs might at first seem limiting for creating applications, it is important to remember that the actual obfuscator can be correctly used on programs that are not necessarily rejecting. The restriction on rejecting programs is needed only in proof steps that invoke the security of the obfuscator and other outside primitives might be used to make other proof steps work. For example, Garg et al. [GGSW13] show how to use witness encryption and unique signatures to give an Identity-Based Encryption scheme. But one where the identity one encrypts to is revealed. If we replace the witness encryption scheme, with an indistinguishability obfuscator for rejecting programs, we get an anonoymous IBE scheme [ABC$^+$05]. [19]

## 8.1 Defining Indistinguishability Obfuscation for Rejecting Programs

We will now define the notion of rejecting indistinguishability obfuscator ($\mathsf{riO}$). The syntax and correctness properties are similar to that of standard indistinguishability obfuscation. The obfuscator is a compiler which takes as input a circuit, and outputs an obfuscated progam which has identical functionality as the input program. For security, we require that if two circuits reject all inputs, then their obfuscations are computationally indistinguishable.

**Syntax** Let $\{\mathcal{C}_\lambda\}_\lambda$ be a circuit family, where each circuit in $\mathcal{C}_\lambda$ has domain $\mathcal{D}_\lambda$ and co-domain $\mathcal{R}_\lambda$. A rejecting indistinguishability obfuscator consists of a compiler $\mathsf{riO.Obf}$ and an evaluator $\mathsf{riO.Eval}$ with the following syntax.

- $\mathsf{riO.Obf}(1^\lambda, C)$ : The obfuscator takes as input a security parameter $\lambda$ and a circuit $C \in \mathcal{C}_\lambda$. It outputs an obfuscated program $\tilde{C}$.
- $\mathsf{riO.Eval}(\tilde{C}, x)$ : The evaluation algorithm takes as input an obfuscated program $\tilde{C}$ and an input $x \in \mathcal{D}_\lambda$. It output $y \in \mathcal{R}_\lambda \cup \{\perp\}$.

**Correctness** Here, we will be considering perfect correctness, which requires that the obfuscated program has identical functionality as the input program. Formally, an obfuscator $\mathsf{riO} = (\mathsf{riO.Obf}, \mathsf{riO.Eval})$ is said to be correct if for all security parameters $\lambda$, circuits $C \in \mathcal{C}_\lambda$ and inputs $x \in \mathcal{D}_\lambda$,

$$\mathsf{riO.Eval}\left(\mathsf{riO.Obf}\left(1^\lambda, C\right), x\right) = C(x).$$

**Security** We will now define the security notion for rejecting indistinguishability obfuscator . This notion guarantees that if two programs reject all inputs, then their obfuscations are computationally indistinguishable.

[19] This resulting construction (perhaps not so surprisingly) is actually very close to what one get if one first constructed an IBE scheme from witness encryption using [GGSW13] and then applied our techniques from Section 5 to hide the identity. If one uses obfuscation for rejecting programs roles these two parts into one.

**Definition 8.1.** An obfuscation scheme $\mathsf{riO} = (\mathsf{riO.Obf}, \mathsf{riO.Eval})$ for circuit class $\{\mathcal{C}_\lambda\}_\lambda$ is said to be a secure rejecting indistinguishability obfuscator if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that the following function (of $\lambda$) is bounded by $\mathrm{negl}(\cdot)$:

$$\left|\Pr\left[\mathcal{A}_1(\tilde{C}_b, \mathsf{st}) = b \ \wedge \ (\forall x \in \mathcal{D}_\lambda,\ C_0(x) = C_1(x) = 0) \ : \ \begin{array}{c}(C_0, C_1, \mathsf{st}) \leftarrow \mathcal{A}_0(1^\lambda), C_b \in \mathcal{C}_\lambda \\ b \leftarrow \{0,1\}, \tilde{C}_b \leftarrow \mathsf{riO.Obf}(1^\lambda, C_b)\end{array}\right] - \frac{1}{2}\right|$$

## 8.2 Witness Encryption

In this section, we will define the notion of witness encryption, first introduced by [GGSW13]. A witness encryption scheme for an NP language $\mathcal{L}$ consists of an encryption algorithm and a decryption algorithm. Using the encryption algorithm, one can encrypt any message for an NP statement $x$ to compute a ciphertext $\mathsf{ct}$. If there exists a witness $w$ proving that $x \in \mathcal{L}$, then one can use $w$ to decrypt the ciphertext $\mathsf{ct}$. The security guarantee states that if $x \notin \mathcal{L}$, then encryption of message $m_0$ for statement $x$ is indistinguishable from encryption of message $m_1$ for statement $x$. We will now give a formal description of the syntax, correctness and security requirements.

**Syntax** Let $\mathcal{L}$ be an NP language with witness relation $\mathcal{R}(\cdot,\cdot)$ (that is, $x \in \mathcal{L}$ iff there exists a witness $w$ such that $\mathcal{R}(x,w) = 1$). Let $n = n(|x|)$ be a polynomial denoting the length of witness for statement $x$. A witness encryption scheme $\mathsf{WE}$ for language $\mathcal{L}$ and message space $\mathcal{M}$ consists of the following algorithms.

- $\mathsf{WE.Enc}(1^\lambda, x, m)$: The encryption algorithm takes as input security parameter $\lambda$, statement $x$ and message $m \in \mathcal{M}$, and outputs a ciphertext $\mathsf{ct}$.
- $\mathsf{WE.Dec}(\mathsf{ct}, w)$: The decryption algorithm takes as input a ciphertext $\mathsf{ct}$, witness $w$ and outputs $y \in \mathcal{M} \cup \{\perp\}$.

**Correctness** For simplicity, we will define perfect correctness.

**Definition 8.2.** A witness encryption scheme $(\mathsf{WE.Enc}, \mathsf{WE.Dec})$ for language $\mathcal{L}$ and message space $\mathcal{M}$ is said to be (perfectly) correct if, for all security parameters $\lambda$, statements $x \in \mathcal{L}$, witnesses $w$ s.t. $R(x,w) = 1$ and messages $m \in \mathcal{M}$,

$$\mathsf{WE.Dec}(\mathsf{WE.Enc}(1^\lambda, x, m), w) = m.$$

**Security** We will be using the adaptive soundness security definition from the work of Bellare and Hoang [BH15].

**Definition 8.3.** A witness encryption scheme $\mathsf{WE} = (\mathsf{WE.Enc}, \mathsf{WE.Dec})$ for language $\mathcal{L}$ and message space $\mathcal{M}$ is said to be secure if for all security parameters $\lambda$, PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$,

$$\Pr\left[x \notin \mathcal{L} \wedge \mathcal{A}_1(\mathsf{ct}, \mathsf{st}) = b : \begin{array}{c}(x, m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_0(1^\lambda) \\ b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, x, m_b)\end{array}\right] \leq \frac{1}{2} + \mathrm{negl}(\lambda).$$

## 8.3 Construction of Rejecting Indistinguishability Obfuscator from Witness Encrytion

We will show how to construct rejecting indistinguishability obfuscator from witness encryption and lockable obfuscation. Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ be a family of circuits, where each circuit in $\mathcal{C}_\lambda$ has depth $d(\lambda)$, takes $n(\lambda)$ bit inputs, outputs a single bit. First, we need to define an NP relation for our witness encryption scheme. The NP language for our witness encryption scheme will be $\mathsf{CIRCUIT\text{-}SAT} = \{C \in \mathcal{C} \ : \ \exists\ w \text{ s.t. } C(w) = 1\}$. The set of witnesses for any $C \in \mathsf{CIRCUIT\text{-}SAT}$ is $\{w \ : \ C(w) = 1\}$. The tools required for constructing $\mathsf{riO}$ for $\mathcal{C}$ are as follows:

**Program** $Q_{\mathsf{ct},C}$

**Constants**: Ciphertext $\mathsf{ct}$, circuit $C$.
**Inputs:** String $w \in \{0,1\}^n$.

1. If $C(w) = 0$, output $0^{\ell_{\text{out}}}$.
2. Else output $\mathsf{WE.Dec}(\mathsf{ct}, w)$.

Figure 7: Program $Q_{\mathsf{ct},C}$

- A witness encryption scheme $\mathsf{WE} = (\mathsf{WE.Enc}, \mathsf{WE.Dec})$ with message space $\{0,1\}^{\ell_{\text{out}}}$ for NP relation $\mathsf{CIRCUIT\text{-}SAT}$. Let $\ell_{\mathsf{ct}}$ denote the size of the ciphertext. Further, we will assume a canonical family of circuits $\{C^{\lambda}_{\mathsf{WE.Dec}}\}_{\lambda}$ where $C^{\lambda}_{\mathsf{WE.Dec}}(\mathsf{ct}, w) = \mathsf{WE.Dec}(\mathsf{ct}, w)$. The circuit $C^{\lambda}_{\mathsf{WE.Dec}}$ has depth $d_{\mathsf{WE.Dec}}(\lambda)$, takes $\ell_{\mathsf{ct}}(\lambda) + n(\lambda)$ bit inputs, and outputs $\ell_{\text{out}}(\lambda)$ bits.
- An lockable obfuscation scheme $(\mathsf{Obf}, \mathsf{Eval})$ with message space $\{0,1\}$ for circuit class $\mathcal{C}_{n,\ell_{\text{out}},d+d_{\mathsf{WE.Dec}}}$.

We will now describe the obfuscation and evaluation algorithms.

- $\mathsf{riO.Obf}(1^{\lambda}, C)$: The obfuscation algorithm first chooses a uniformly random string $\alpha \leftarrow \{0,1\}^{\ell_{\text{out}}}$. Next, it computes a WE ciphertext $\mathsf{ct} \leftarrow \mathsf{WE.Enc}(1^{\lambda}, C, \alpha)$.[20] Let $Q_{\mathsf{ct},C}(\cdot)$ be the circuit defined in Figure 7. The obfuscator then computes an lockable obfuscation $\tilde{C} \leftarrow \mathsf{Obf}(1^{\lambda}, Q_{\mathsf{ct},C}, 1, \alpha)$. The final obfuscation is $\tilde{C}$.
- $\mathsf{riO.Eval}(\tilde{C}, x)$: The evaluation algorithm first computes $y = \mathsf{Eval}(\tilde{C}, x)$. If $y = 1$, the $\mathsf{riO}$ evaluator outputs 1, else it outputs 0.

**Correctness** We need to show that the obfuscated program has identical functionality as the original (input) program. Fix any security parameter $\lambda$, circuit $C \in \mathcal{C}_{\lambda}$ and input $x \in \{0,1\}^n$. Let $\alpha$ be the random string chosen during obfuscation, $\mathsf{ct} \leftarrow \mathsf{WE.Enc}(1^{\lambda}, C, \alpha)$ and $\tilde{C} \leftarrow \mathsf{Obf}(1^{\lambda}, 1, \alpha)$.

If $C(x) = 0$, then $Q_{\mathsf{ct},C}(x) = 0^{\ell_{\text{out}}}$. As a result, $\mathsf{Eval}(\tilde{C}, x) = \perp$ with all but negligible probability over the coins chosen during lockable obfuscation (assuming the semi-statistical correctness of lockable obfuscation scheme). Hence, $\mathsf{riO.Eval}(\tilde{C}, x) = 0$ with all but negligible probability.

If $C(x) = 1$, then $Q_{\mathsf{ct},C}(w) = \alpha$ (assuming perfect correctness of the witness encryption scheme), and hence $\mathsf{riO.Eval}(\tilde{C}, x) = \mathsf{Eval}(\tilde{C}, x) = 1$ (assuming correctness of lockable obfuscation scheme).

### 8.3.1 Security

We will prove security by defining a sequence of computationally indistinguishable hybrids, and finally showing that in the last hybrid, the adversary has zero advantage. At a high level, our proof works as follows. Recall that the adversary must output two programs that reject all inputs. As a result, if we encrypt a string $\alpha$ to either of these programs using the witness encryption scheme, then $\alpha$ is computationally hidden. Hence, we can replace $\alpha$ with the all-zeroes string. Having done this, we can now use the security of our lockable obfuscation scheme to simulate the obfuscated program.

We will now formally define the hybrid experiments.

**Hybrid** $H_0$ : This corresponds to the real experiment.

1. The challenger receives as input two circuits $C_0, C_1$ such that both circuits output 0 on all inputs.
2. The challenger chooses a uniformly random string $\alpha \leftarrow \{0,1\}^{\ell_{\text{out}}}$ and $b \leftarrow \{0,1\}$.
3. It then computes $\mathsf{ct} \leftarrow \mathsf{WE.Enc}(1^{\lambda}, C_b, \alpha)$.
4. Finally, it computes $\tilde{C} \leftarrow \mathsf{Obf}(1^{\lambda}, Q_{\mathsf{ct},C_b}, 1, \alpha)$ and sends $\tilde{C}$ to the adversary. The adversary outputs a bit $b'$ and wins if $b = b'$.

[20] By circuit $C$, we represent the statement that $\exists\ w$ such that $C(w) = 1$.

**Hybrid** $H_1$ : This hybrid is similar to the previous one, except that the challenger encrypts $0^{\ell_{\text{out}}}$ instead of $\alpha$.

3. It then computes $\mathsf{ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, C_b, 0^{\ell_{\text{out}}})$.

**Hybrid** $H_2$ : In this hybrid experiment, the challenger uses the simulator for the lockable obfuscation scheme to output the final obfuscation. Let $\mathsf{Sim}$ be the simulator for the lockable obfuscation scheme. Let $s = |Q_{\mathsf{ct},C_0}| = |Q_{\mathsf{ct},C_1}|$.

4. Finally, it computes $\tilde{C} \leftarrow \mathsf{Sim}(1^\lambda, 1^s)$ and sends $\tilde{C}$ to the adversary. The adversary outputs a bit $b'$ and wins if $b = b'$.

Let $\mathsf{Adv}^i_{\mathcal{A}}$ denote the advantage of adversary $\mathcal{A}$ in hybrid $H_i$.

**Claim 8.1.** Assuming the witness encryption scheme $\mathsf{WE}$ is secure, for any PPT adversary $\mathcal{A}$, $|\mathsf{Adv}^0_{\mathcal{A}} - \mathsf{Adv}^1_{\mathcal{A}}| \leq \text{negl}(\lambda)$.

*Proof.* Suppose, on the contrary, there exists a PPT adversary $\mathcal{A}$ such that $|\mathsf{Adv}^0_{\mathcal{A}} - \mathsf{Adv}^1_{\mathcal{A}}| = \epsilon$. We will use $\mathcal{A}$ to construct a reduction algorithm $\mathcal{B}$ that breaks the security of $\mathsf{WE}$.

The adversary first sends two circuits $C_0, C_1$. The reduction algorithm chooses a string $\alpha \leftarrow \{0,1\}^{\ell_{\text{out}}}$, $b \leftarrow \{0,1\}$ and sends $C_b, \alpha, 0^{\ell_{\text{out}}}$ to the witness encryption challenger. It receives a ciphertext $\mathsf{ct}$. It then computes $\tilde{C} \leftarrow \mathsf{Obf}(1^\lambda, Q_{\mathsf{ct},C}, 1, \alpha)$ and sends $\tilde{C}$ to the adversary. The adversary sends its guess $b'$. If $b = b'$, the reduction algorithm guesses that $\alpha$ was encrypted, else it guesses that $0^{\ell_{\text{out}}}$ was encrypted.

■

**Claim 8.2.** Assuming the security of lockable obfuscation scheme, for any PPT adversary $\mathcal{A}$, $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}| \leq \text{negl}(\lambda)$.

*Proof.* Suppose, on the contrary, there exists a PPT adversary $\mathcal{A}$ such that $|\mathsf{Adv}^0_{\mathcal{A}} - \mathsf{Adv}^1_{\mathcal{A}}| = \epsilon$. We will use $\mathcal{A}$ to construct a reduction algorithm $\mathcal{B}$ that breaks the security of the lockable obfuscation scheme.

The adversary sends two circuits $C_0, C_1$. The reduction algorithm first chooses $b \leftarrow \{0,1\}$ and computes an encryption of $0^{\ell_{\text{out}}}$ for $C_b$; that is, it computes $\mathsf{ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, C_b, 0^{\ell_{\text{out}}})$. It then sends circuit $Q_{\mathsf{ct},C_b}$ and message 1 to the lockable obfuscator challenger, and receives an obfuscated program $\tilde{C}$ which it forwards to $\mathcal{A}$. If $\mathcal{A}$ guesses correctly, then $\mathcal{B}$ guesses that $\tilde{C}$ is an honestly computed obfuscation, else it guesses that $\tilde{C}$ is simulated.

■

**Claim 8.3.** For any adversary $\mathcal{A}$, $|\mathsf{Adv}^2_{\mathcal{A}}| = 0$.

*Proof.* This follows directly from the definition of hybrid $H_2$. Note that the challenger only requires the size of $Q_{\mathsf{ct},C_b}$, and this is independent of $b$ since $|C_0| = |C_1|$. As a result, the final obfuscated program $\tilde{C}$ contains no information about the bit $b$. ■

# 9 Upgrading Broadcast Encryption to Anonymous Broadcast Encryption

The notion of broadcast encryption was first formally discussed in [FN94]. In a broadcast encryption scheme, we have a set of $N$ users. There is a common public key for all the $N$ users, and each user has a private decryption key. Using the encryption algorithm, one can encrypt a message for any subset of users. Suppose message $m$ is encrypted for set $S$. Then any person in set $S$ must be able to recover $m$ using the decryption key, and if person $i$ is not in the set $S$, then person $i$ should not learn the message $m$. For

efficiency/succinctness, we require that the size of the ciphertext be at most $|S| + \mathsf{poly}(\lambda)$, where $\lambda$ is the security parameter.

While hiding the message might suffice for certain applications, one might be interested in the case where even the set $S$ is hidden. This is referred to as anonymous/recipient-private broadcast encryption, introduced by [BBW06]. In this work, we will consider a weaker notion which we call *one-sided* anonymous broadcast encryption. Here, if a message is encrypted for a set $S$, then for any user $i \notin S$, both the message and the set $S$ will be hidden.

In this section, we will first define the notions of broadcast encryption and one-sided anonymous broadcast encryption, and then show how to transform any broadcast encryption scheme to a one-sided anonymous broadcast encryption scheme.

## 9.1 Preliminaries

We will first introduce the syntax, correctness and security properties of broadcast encryption. A broadcast encryption scheme $\mathsf{BE}$ with message space $\mathcal{M}$ consists of the following algorithms.

- $\mathsf{Setup}(1^\lambda, N)$: The setup algorithm takes as input the security parameter $\lambda$ and a bound $N$ on the total number of users. It outputs a pubic key $\mathsf{pk}$ and $N$ decryption keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$.
- $\mathsf{Encrypt}(\mathsf{pk}, m \in \mathcal{M}, S \subseteq [N])$: The encryption algorithm takes as input the public key $\mathsf{pk}$, a message $m \in \mathcal{M}$ and a subset $S \subseteq [N]$ . It outputs a ciphertext $\mathsf{ct}$.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input a secret key $\mathsf{sk}$ and a ciphertext $\mathsf{ct}$, and outputs $y \in \mathcal{M} \cup \{\perp\}$.

**Correctness** We require that for all security parameters $\lambda$ and bound $N$ on total number of users, for all sets $S \subseteq [N]$, messages $m \in \mathcal{M}$, $(\mathsf{pk}, \{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$ and $\mathsf{ct} \leftarrow \mathit{Encrypt}(\mathsf{pk}, m, S)$, if $i \in S$, then $\mathsf{Decrypt}(\mathsf{sk}_i, \mathsf{ct}) = m$.

**Succinctness** For succinctness, we require that for all $\lambda, N$, $(\mathsf{pk}, \{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\}) \leftarrow \mathsf{Setup}(1^\lambda, N)$, messages $m \in \mathcal{M}$ and $S \subseteq [N]$, if $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m, S)$ then $|\mathsf{ct}| \leq \mathsf{poly}(\lambda) + |S|$.

### 9.1.1 Security

We will now present two different notions of security. The first one only hides the message encrypted, while the second notion requires that the set of users associated with a ciphertext is also hidden.

**Definition 9.1.** Let $\mathsf{BE} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a broadcast encryption scheme. Consider the following security game between a challenger and an adversary $\mathcal{A}$.

- **Setup Phase** The challenger chooses $(\mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_N)) \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$ and sends $\mathsf{pk}$ to $\mathcal{A}$.
- **Pre-Challenge Query Phase** In this phase, the adversary queries for secret keys. It sends an index $i \in [N]$, and receives the secret key $\mathsf{sk}_i$.
- **Challenge Phase** The adversary sends two challenge messages $m_0, m_1$ and a set $S$. The challenger chooses $b \leftarrow \{0, 1\}$, computes $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m_b, S)$ and sends $\mathsf{ct}$ to the challenger.
- **Post-Challenge Query Phase** This is similar to the pre-challenge query phase. In this phase, the adversary sends an index $i \in [N]$, and receives the secret key $\mathsf{sk}_i$.
- **Guess** The adversary finally sends its guess $b'$. Let $Q$ be the set of indices queried by $\mathcal{A}$. The adversary wins if $b = b'$ and $Q \cap S = \phi$.

The scheme is said to be secure if for all security parameters $\lambda$ and all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the above security game is at most negligible.

**Definition 9.2** (One-sided anonymous broadcast encryption)**.** Let $\mathsf{BE} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a broadcast encryption scheme. Consider the following security game between a challenger and an adversary $\mathcal{A}$.

- **Setup Phase** The challenger chooses $(\mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_N)) \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$ and sends $\mathsf{pk}$ to $\mathcal{A}$.
- **Pre-Challenge Query Phase** In this phase, the adversary queries for secret keys. It sends an index $i \in [N]$, and receives the secret key $\mathsf{sk}_i$.
- **Challenge Phase** The adversary sends two challenge messages $m_0, m_1$ and two sets $S_0, S_1$. The challenger chooses $b \leftarrow \{0,1\}$, computes $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m_b, S_b)$ and sends $\mathsf{ct}$ to the challenger.
- **Post-Challenge Query Phase** This is similar to the pre-challenge query phase. In this phase, the adversary sends an index $i \in [N]$, and receives the secret key $\mathsf{sk}_i$.
- **Guess** The adversary finally sends its guess $b'$. Let $Q$ be the set of indices queried by $\mathcal{A}$. The adversary wins if $b = b'$ and $Q \cap (S_0 \cup S_1) = \phi$.

The scheme is said to be secure if for all security parameters $\lambda$ and all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the above security game is at most negligible.

## 9.2 Upgrading Broadcast Encryption to One-Sided Anonymous Broadcast Encryption

In this section, we will show how to upgrade any broadcast encryption scheme to achieve one-sided anonymity. Let $\mathsf{BE} = (\mathsf{BE.Setup}, \mathsf{BE.Encrypt}, \mathsf{BE.Decrypt})$ be a broadcast encryption scheme with message space $\{0,1\}^*$. Let $\ell_{\mathsf{ct}}(\lambda)$ denote the length of ciphertext corresponding to message $m \in \{0,1\}^\lambda$. Our scheme with one-sided anonymity $\mathsf{aBE} = (\mathsf{aBE.Setup}, \mathsf{aBE.Encrypt}, \mathsf{aBE.Decrypt})$ uses the following cryptographic primitives:

- PRF $F$ with keyspace $\mathcal{K}$, input space $\{0,1\}^\lambda$ and output space $\{0,1\}$. Let $d_F$ denote depth of circuit evaluating $F(\cdot,\cdot)$.
- Leveled homomorphic encryption scheme $\mathsf{LHE} = (\mathsf{LHE.Setup}, \mathsf{LHE.Enc}, \mathsf{LHE.Dec}, \mathsf{LHE.Eval})$ with decryption circuit having depth bounded by $d_{\mathsf{LHE}}(\cdot)$.
- Lockable obfuscation scheme $(\mathsf{Obf}, \mathsf{Eval})$

The scheme can be described as follows.

- $\mathsf{aBE.Setup}(1^\lambda, 1^N)$: The setup algorithm computes $(\mathsf{pk}, \{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\}) \leftarrow \mathsf{BE.Setup}(1^\lambda, 1^N)$ and outputs $\mathsf{pk}$ as the public key and $\{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\}$ as the secret keys.
- $\mathsf{aBE.Encrypt}(\mathsf{pk}, m, S)$: The encryption algorithm first chooses a random string $\alpha \leftarrow \{0,1\}^\lambda$ and computes $\mathsf{ct}' \leftarrow \mathsf{BE.Encrypt}(\mathsf{pk}, \alpha, S)$. Next, it chooses a PRF key $K \leftarrow \mathcal{K}$ and LHE keys $(\mathsf{lhe.pk}, \mathsf{lhe.sk}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^{d_F})$. It computes $\mathsf{mask} = (F(K,1), F(K,2), \ldots, F(K, \ell_{\mathsf{ct}}))$ and sets $\mathsf{ct}_1 = \mathsf{ct}' \oplus \mathsf{mask}$. Let $\mathsf{ct}_2 \leftarrow \mathsf{LHE.Enc}(\mathsf{lhe.pk}, K)$. Finally, let $C_{\mathsf{LHE.Dec}}$ be the LHE decryption circuit with $\mathsf{lhe.sk}$ hardwired. The encryption algorithm computes $P \leftarrow \mathsf{Obf}(C_{\mathsf{LHE.Dec}}, \alpha, m)$. The final ciphertext is $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, P, \mathsf{lhe.pk})$.
- $\mathsf{aBE.Decrypt}(\mathsf{sk}, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, P, \mathsf{lhe.pk}))$: The decryption algorithm first computes an LHE evaluation using the encryption of PRF key. Consider a circuit $C_F$ that takes as input a PRF key $K$ and outputs $(F(K,1), F(K,2), \ldots, F(K, \ell_{\mathsf{ct}}))$. The decryption algorithm performs LHE evaluation on $\mathsf{ct}_2$ with $C_F$. It computes $\widetilde{\mathbf{ct}}_2 = \mathsf{LHE.Eval}(\mathsf{ct}_2, C_F)$. Next, let $C_{\mathsf{ct}_1 \oplus}$ denote the function that has $\mathsf{ct}_1$ hardwired, takes input $y$ and outputs $\mathsf{ct}_1 \oplus y$. The decryption algorithm computes $\widetilde{\mathbf{ct}}_1 = \mathsf{LHE.Eval}(\widetilde{\mathbf{ct}}_2, C_{\mathsf{ct}_1 \oplus})$.

  Next, consider the circuit $C_{\mathsf{BE.Decrypt}} = \mathsf{BE.Decrypt}(\mathsf{sk}, \cdot)$ which has a secret key $\mathsf{sk}$ hardwired. Let $\widetilde{\mathbf{ct}} = \mathsf{LHE.Eval}(\widetilde{\mathbf{ct}}_1, C_{\mathsf{BE.Decrypt}})$. Finally, the decryption algorithm outputs $y = \mathsf{Eval}(P, \widetilde{\mathbf{ct}})$.

**Correctness** Fix any security parameter $\lambda$, $N \in \mathbb{N}$, message $m \in \{0,1\}^\lambda$ and set $S \subseteq [N]$. Let $(\mathsf{pk}, \{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\}) \leftarrow \mathsf{aBE.Setup}(1^\lambda, N)$ be the public/secret keys. The encryption algorithm chooses $\alpha \leftarrow \{0,1\}^\lambda$, $K \leftarrow \mathcal{K}$, $(\mathsf{lhe.pk}, \mathsf{lhe.sk}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^{d_F})$ and sets $\mathsf{ct}' \leftarrow \mathsf{BE.Encrypt}(\mathsf{pk}, \alpha, S)$, $\mathsf{ct}_2 = \mathsf{LHE.Enc}(\mathsf{lhe.pk}, K)$. Finally, it computes $P \leftarrow \mathsf{Obf}(C_{\mathsf{LHE.Dec}}, \alpha, m)$.

The decryption algorithm first computes $\widetilde{\mathbf{ct}}_2 = \mathsf{LHE.Eval}(\mathsf{ct}_2, C_F)$. Note that $\widetilde{\mathbf{ct}}_2$ is an LHE encryption of $\mathsf{mask} = (F(K,1), \ldots, F(K,\ell_{\mathsf{ct}}))$. Next, the decryption algorithm computes $\widetilde{\mathbf{ct}}_1 = \mathsf{LHE.Eval}(\widetilde{\mathbf{ct}}_2, C_{\mathsf{ct}_1 \oplus})$. By construction, it follows that $\mathsf{BE.Decrypt}(\mathsf{sk}, \mathsf{LHE.Dec}(\mathsf{lhe.sk}, \widetilde{\mathbf{ct}}_1)) = \alpha$. Therefore, $\widetilde{\mathbf{ct}} = \mathsf{LHE.Eval}(\widetilde{\mathbf{ct}}_1, C_{\mathsf{BE.Decrypt}})$ is an LHE encryption of $\alpha$. As a result, the obfuscated program $P$, when evaluated with input $\widetilde{\mathbf{ct}}$, outputs $m$.

**Efficiency** The ciphertext in our anonymous broadcast encryption scheme consists of three components. The first component $\mathsf{ct}_1$ has size $\ell_{\mathsf{ct}}(\lambda) = \mathsf{poly}_1(\lambda) + N$. This follows from the efficiency of the underlying broadcast encryption scheme. Next, the second component is an LHE encryption of PRF key $K$, and therefore its size is bounded by $\mathsf{poly}_2(\lambda)$, independent of $N$. Finally, the last component is an obfuscation of LHE decryption circuit. This also depends only on the security parameter, and is independent of $N$. As a result, the ciphertext output by our anonymous broadcast encryption scheme has size $\mathsf{poly}(\lambda) + N$.

### 9.2.1 Security

In this section, we will show that the scheme described above satisfies the one-sided anonymity definition (see Definition 9.2). We will prove security via a sequence of hybrid experiments $H_0, H_1$, where $H_0$ corresponds to the original security game.

**Hybrid** $H_0$

- **Setup Phase** The challenger chooses $(\mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_N)) \leftarrow \mathsf{BE.Setup}(1^\lambda, 1^N)$ and sends $\mathsf{pk}$ to $\mathcal{A}$.
- **Pre-Challenge Query Phase** In this phase, the adversary queries for secret keys. It sends an index $i \in [N]$, and receives the secret key $\mathsf{sk}_i$.
- **Challenge Phase** The adversary sends two challenge messages $m_0, m_1$ and two sets $S_0, S_1$. The challenger chooses $b \leftarrow \{0,1\}$.
  1. It chooses $\alpha \leftarrow \{0,1\}^\lambda$ and computes $\mathsf{ct}' \leftarrow \mathsf{BE.Encrypt}(\mathsf{pk}, \alpha, S_b)$.
  2. Next it chooses a PRF key $K$ and sets $\mathsf{mask} = (F(K,1), F(K,2), \ldots, F(K,\ell_{\mathsf{ct}}))$, $\mathsf{ct}_1 = \mathsf{ct}' \oplus \mathsf{mask}$.
  3. It then chooses LHE keys $(\mathsf{lhe.pk}, \mathsf{lhe.sk}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^{d_F})$ and sets $\mathsf{ct}_2 \leftarrow \mathsf{LHE.Enc}(\mathsf{lhe.pk}, K)$.
  4. Finally, let $C_{\mathsf{LHE.Dec}}$ be the LHE decryption circuit with $\mathsf{lhe.sk}$ hardwired. The challenger computes $P \leftarrow \mathsf{Obf}(C_{\mathsf{LHE.Dec}}, \alpha, m_b)$ and outputs $(\mathsf{ct}_1, \mathsf{ct}_2, P)$.
- **Post-Challenge Query Phase** This is similar to the pre-challenge query phase. In this phase, the adversary sends an index $i \in [N]$, and receives the secret key $\mathsf{sk}_i$.
- **Guess** The adversary finally sends its guess $b'$. Let $Q$ be the set of indices queried by $\mathcal{A}$. The adversary wins if $b = b'$ and $Q \cap (S_0 \cup S_1) = \phi$.

**Hybrid** $H_1$ This hybrid is identical to the previous one, except that the challenger computes a broadcast encryption of $\mathbf{0}$ instead of the lock $\alpha$.

1. It chooses $\alpha \leftarrow \{0,1\}^\lambda$ and computes $\mathsf{ct}' \leftarrow \mathsf{BE.Encrypt}(\mathsf{pk}, \mathbf{0}, S_b)$.

**Hybrid** $H_2$ In this hybrid experiment, the challenger simulates the obfuscated program. Instead of computing $P \leftarrow obf(C_{\mathsf{LHE.Dec}}, \alpha, m_b)$, it computes $P \leftarrow \mathsf{Sim}(1^{|C_{\mathsf{LHE.Dec}}|}, 1^{|m_b|})$.

4. Finally, let $C_{\mathsf{LHE.Dec}}$ be the LHE decryption circuit with $\mathsf{lhe.sk}$ hardwired. The challenger computes $P \leftarrow \mathsf{Sim}(1^{|C_{\mathsf{LHE.Dec}}|}, 1^{|m_b|})$ and outputs $(\mathsf{ct}_1, \mathsf{ct}_2, P)$.

**Hybrid** $H_3$ In this hybrid experiment, the challenger replaces the LHE encryption of PRF key $K$ with encryption of $\mathbf{0}$.

3. It then chooses LHE keys $(\mathsf{lhe.pk}, \mathsf{lhe.sk}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^{d_F})$ and sets $\mathsf{ct}_2 \leftarrow \mathsf{LHE.Enc}(\mathsf{lhe.pk}, \mathbf{0})$.

**Hybrid** $H_4$ This experiment is similar to the previous one, except that the challenger sets $\mathsf{ct}_1$ to be a uniformly random string.

2. Next it chooses $\mathsf{ct}_1 \leftarrow \{0,1\}^{\ell_{\mathsf{ct}}}$.

**Analysis** For any PPT adversary $\mathcal{A}$, let $\mathsf{Adv}^i_{\mathcal{A}}$ denote the advantage of $\mathcal{A}$ in the hybrid experiment $H_i$.

**Claim 9.1.** Assuming $\mathsf{BE}$ is a secure broadcast encryption scheme, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^0_{\mathcal{A}} - \mathsf{Adv}^1_{\mathcal{A}} \leq \mathrm{negl}(\lambda)$.

*Proof.* The only difference between these two hybrids is with regard to the ciphertext component $\mathsf{ct}'$. In hybrid $H_0$, the challenger computes a broadcast encryption of $\alpha$, while in $H_1$, it computes an encryption of $\mathbf{0}$. Also, note that the adversary's queries lie outside the set $S_0 \cup S_1$. Suppose there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^0_{\mathcal{A}} - \mathsf{Adv}^1_{\mathcal{A}} = \epsilon$. We will use $\mathcal{A}$ to construct a reduction algorithm $\mathcal{B}$ that breaks the security of $\mathsf{BE}$.

The reduction algorithm $\mathcal{B}$ first receives the public key from the challenger, which it then forwards to the adversary. Next, it receives secret key queries from the adversary, which it forwards to the challenger. For each key query $i$, the challenger sends $\mathsf{sk}_i$ to $\mathcal{B}$, and the reduction algorithm forwards it to $\mathcal{A}$. Next, the adversary sends two challenge messages $m_0, m_1$ together with sets $S_0, S_1$. The reduction algorithm first chooses a bit $b \leftarrow \{0,1\}$. It then sends $m_b, \mathbf{0}$ as the challenge messages, and $S_b$ as the challenge set. The challenger sends back the challenge ciphertext $\mathsf{ct}'$. The reduction algorithm then chooses a PRF key $K$, LHE keys $(\mathsf{lhe.pk}, \mathsf{lhe.sk}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^{d_F})$, sets $\mathsf{ct}_1 = \mathsf{ct}' \oplus (F(K,1), \ldots, F(K, \ell_{\mathsf{ct}}))$, $\mathsf{ct}_2 \leftarrow \mathsf{LHE.Enc}(\mathsf{lhe.pk}, K)$ and $P \leftarrow \mathsf{Obf}(C_{\mathsf{LHE.Dec}}, \alpha, m_b)$. It sends $(\mathsf{ct}_1, \mathsf{ct}_2, P)$ to $\mathcal{A}$. Next, the adversary sends post-challenge queries which are handled similar to the pre-challenge ones. Finally, if the adversary's guess is correct, the reduction algorithm guesses that $\mathsf{ct}'$ is encryption of $\mathbf{0}$, else it guesses that $\mathsf{ct}'$ is encryption of $m_b$.

Depending on whether the challenger sends an encryption of $m_b$ or $\mathbf{0}$, the reduction algorithm simulates either hybrid experiment $H_0$ or $H_1$. ∎

**Claim 9.2.** Assuming $(\mathsf{Obf}, \mathsf{Eval})$ is a secure lockable obfuscation scheme, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}} \leq \mathrm{negl}(\lambda)$.

*Proof.* Suppose, on the contrary, there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}} = \epsilon$ for some non-negligible function $\epsilon$. We will use $\mathcal{A}$ to build a reduction algorithm $\mathcal{B}$ that breaks the security of lockable obfuscation scheme.

The reduction algorithm first chooses the public/secret keys of broadcast encryption scheme and sends the public key to $\mathcal{A}$. Next, it handles the queries of the adversary using the secret keys. In the challenge phase, the adversary sends $m_0, m_1$ and subsets $S_0, S_1$. The reduction algorithm chooses $b \leftarrow \{0,1\}$ and computes $\mathsf{ct}' \leftarrow \mathsf{BE.Encrypt}(\mathsf{pk}, \mathbf{0}, S_b)$. It chooses PRF key $K$, LHE keys $(\mathsf{lhe.pk}, \mathsf{lhe.sk})$ and sets $\mathsf{ct}_1 = \mathsf{ct}' \oplus (F(K,1), \ldots, F(K, \ell_{\mathsf{ct}}))$ and $\mathsf{ct}_2 \leftarrow \mathsf{LHE.Enc}(\mathsf{lhe.pk}, \mathbf{K})$. Next, it sends $C_{\mathsf{LHE.Dec}}$ and message $m_b$ to the challenger and receives an obfuscated program $P$. It sends $(\mathsf{ct}_1, \mathsf{ct}_2, P)$ to $\mathcal{A}$. The adversary finally sends its guess. If the guess is correct, the reduction algorithm guesses that $P$ is a simulated obfuscation, else it guesses that $P$ is an honestly computed obfuscation.

Note that the lock used by the obfuscator is not used anywhere else in hybrids $H_1$ and $H_2$. As a result, the simulator can correctly simulate either $H_1$ or $H_2$ depending on whether $P$ is simulated or not. ∎

**Claim 9.3.** Assuming $\mathsf{LHE}$ is a secure leveled homomorphic encryption scheme, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}} \leq \mathrm{negl}(\lambda)$.

*Proof.* The only difference between these two hybrids is that in one case, the challenge ciphertext has an LHE encryption of the PRF key $K$, while in the other case, it is replaced by an encryption of $\mathbf{0}$. Suppose there exists an adversary that can distinguish between the two hybrids with advantage $\epsilon$. We will use this adversary to break the INDCPA security of $\mathsf{LHE}$.

The reduction algorithm first chooses the public/secret keys of broadcast encryption scheme and sends the public key to $\mathcal{A}$. Next, it handles the queries of the adversary using the secret keys. In the challenge phase, the adversary sends $m_0, m_1$ and subsets $S_0, S_1$. The reduction algorithm chooses $b \leftarrow \{0,1\}$ and computes $\mathsf{ct}' \leftarrow \mathsf{BE.Encrypt}(\mathsf{pk}, \mathbf{0}, S_b)$. It chooses PRF key $K$, LHE keys $(\mathsf{lhe.pk}, \mathsf{lhe.sk})$ and sets $\mathsf{ct}_1 = \mathsf{ct}' \oplus (F(K,1), \ldots, F(K, \ell_{\mathsf{ct}}))$.

Next, the reduction algorithm sends $K, \mathbf{0}$ to the challenger. It receives an LHE public key $\mathsf{lhe.pk}$ and ciphertext $\mathsf{ct}_2$ from the challenger. Next, it computes $P \leftarrow \mathsf{Sim}(1^{|C_{\mathsf{LHE.Dec}}|}, 1^{|m_b|})$ and sends $(\mathsf{ct}_1, \mathsf{ct}_2, P, \mathsf{lhe.pk})$ to $\mathcal{A}$. The adversary finally sends its guess. If the guess is correct, the reduction algorithm guesses that $\mathsf{ct}_2$ is an encryption of $\mathbf{0}$, else it guesses that it is an encryption of $K$.

Note that since the obfuscated program $P$ is simulated, the reduction algorithm does not require the LHE secret key. ∎

**Claim 9.4.** Assuming $F$ is a secure pseudorandom function, for any adversary $\mathcal{A}$, $\mathsf{Adv}^3_{\mathcal{A}} - \mathsf{Adv}^4_{\mathcal{A}} \leq \mathrm{negl}(\lambda)$.

*Proof.* The only difference between these two hybrids is that in one case, $\mathsf{mask} = (F(K,1), \ldots, F(K, \ell_{\mathsf{ct}}))$, while in the other case, $\mathsf{mask}$ is uniformly random. Note that the PRF key is not required anywhere else in the security game (in particular, recall that $\mathsf{ct}_2$ is an encryption of $\mathbf{0}$). As a result, it follows directly from the security of PRF $F$ that no PPT adversary can distinguish between these two hybrids with advantage greater than $\mathrm{negl}(\lambda)$. ∎

**Claim 9.5.** For an adversary $\mathcal{A}$, $\mathsf{Adv}^4_{\mathcal{A}} = 0$.

*Proof.* In hybrid $H_4$, the ciphertext component $\mathsf{ct}_1$ is a uniformly random string, $\mathsf{ct}_2$ is an LHE encryption of $\mathbf{0}$ and $P$ is a simulated program which is independent of the bit $b$ chosen by challenger. As a result, any adversary $\mathcal{A}$ has zero advantage in this experiment. ∎

**Acknowledgements.** We thank Mihir Bellare and Joseph Jaeger for their detailed and informative feedback on our definitions and proofs.

# References

[ABBC10] Tolga Acar, Mira Belenkiy, Mihir Bellare, and David Cash. Cryptographic agility and its relation to circular encryption. In *EUROCRYPT '10*, volume 6110 of LNCS, pages 403–422. Springer, 2010.

[ABC+05] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In *Annual International Cryptology Conference*, 2005.

[ABHS09] Pedro Adão, Gergei Bana, Jonathan Herzog, and Andre Scedrov. Soundness and completeness of formal encryption: The cases of key cycles and partial information leakage. *Journal of Computer Security*, 17(5):737–797, 2009.

[ABN08] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. Cryptology ePrint Archive, Report 2008/440, 2008.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618, 2009.

[ADGM16] Daniel Apon, Nico Döttling, Sanjam Garg, and Pratyay Mukherjee. Cryptanalysis of indistinguishability obfuscations of circuits over ggh13. Cryptology ePrint Archive, Report 2016/1003, 2016.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 308–326, 2015.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Achieving compactness generically: Indistinguishability obfuscation from non-compact functional encryption. *IACR Cryptology ePrint Archive*, 2015.

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings*, pages 1–9, 1999.

[AP16] Navid Alamati and Chris Peikert. Three's compromised too: Circular insecurity for any cycle length from (ring-)lwe. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 659–680, 2016.

[AS16] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *EUROCRYPT*, 2016.

[Att14] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 557–577, 2014.

[Bar86] D A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, STOC '86, 1986.

[BBC+14] Boaz Barak, Nir Bitansky, Ran Canetti, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Obfuscation for evasive functions. In *Theory of Cryptography Conference*, 2014.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 566–582. Springer, 2001.

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam ONeill. Deterministic and efficiently searchable encryption. In *Annual International Cryptology Conference*, 2007.

[BBW06] Adam Barth, Dan Boneh, and Brent Waters. Privacy in encrypted content distribution using private broadcast encryption. In *Financial Cryptography and Data Security, 10th International Conference, FC 2006, Anguilla, British West Indies, February 27-March 2, 2006, Revised Selected Papers*, pages 52–64, 2006.

[BDFP86] Allan Borodin, Danny Dolev, Faith E. Fich, and Wolfgang J. Paul. Bounds for width two branching programs. *SIAM J. Comput.*, 15(2):549–560, 1986.

[BFM15] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Random-oracle uninstantiability from indistinguishability obfuscation. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, 2015.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 533–556, 2014.

[BGH+15] Zvika Brakerski, Craig Gentry, Shai Halevi, Tancrède Lepoint, Amit Sahai, and Mehdi Tibouchi. Cryptanalysis of the quadratic zero-testing of GGH. *IACR Cryptology ePrint Archive*, 2015.

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

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

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, 2014.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.

[BH15] Mihir Bellare and Viet Tung Hoang. Adaptive witness encryption and asymmetric password-based cryptography. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pages 308–331, 2015.

[BHSV98] Mihir Bellare, Shai Halevi, Amit Sahai, and Salil Vadhan. Many-to-one trapdoor functions and their relation to public-key cryptosystems. In *Annual International Cryptology Conference*, 1998.

[BHW15] Allison Bishop, Susan Hohenberger, and Brent Waters. New circular security counterexamples from decision linear and learning with errors. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 776–800, 2015.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 575–584, 2013.

[BMSZ16] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, 2016.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 719–737, 2012.

[BR14] Zvika Brakerski and Guy N Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *Theory of Cryptography Conference*, 2014.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *FOCS*, pages 97–106, 2011.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 171–190, 2015.

[BVWW16] Zvika Brakerski, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Obfuscating conjunctions under entropic ring lwe. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, 2016.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *Proceedings of the 4th conference on Theory of cryptography*, TCC'07, pages 535–554, Berlin, Heidelberg, 2007. Springer-Verlag.

[BWZ14] Dan Boneh, David J. Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. Cryptology ePrint Archive, Report 2014/930, 2014.

[CC09] Melissa Chase and Sherman S. M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *ACM Conference on Computer and Communications Security*, pages 121–130, 2009.

[CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained prfs for nc1 from lwe. In *EUROCRYPT*, 2017.

[CFL+16] Jung Hee Cheon, Pierre-Alain Fouque, Changmin Lee, Brice Minaud, and Hansol Ryu. Cryptanalysis of the new clt multilinear map over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2016.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *STOC*, pages 209–218, 1998.

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

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, 2015.

[CH85] Stephen A. Cook and H. James Hoover. A depth-universal circuit. *SIAM Journal on Computing*, 14(4):833–839, 1985.

[Cha07] Melissa Chase. Multi-authority attribute based encryption. In *TCC*, pages 515–534, 2007.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 3–12, 2015.

[CJL16] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low-level encoding of zero. *LMS Journal of Computation and Mathematics*, 2016.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *IACR Cryptology ePrint Archive*, 2001:19, 2001.

[CLLT16] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of GGH15 multilinear maps. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, 2016.

[CLLT17] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Zeroizing attacks on indistinguishability obfuscation over CLT13. In *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I*, 2017.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 476–493, 2013.

[CLT14] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. Cryptology ePrint Archive, Report 2014/975, 2014.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, 2015.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 523–540, 2004.

[DS05] Yevgeniy Dodis and Adam D. Smith. Correcting errors without leaking partial information. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 654–663, 2005.

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '93, pages 480–491, 1994.

[FO99a] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In *International Workshop on Public Key Cryptography*, pages 53–68. Springer, 1999.

[FO99b] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO '99*, volume 1666 of LNCS, pages 537–554. Springer, 1999.

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

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

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indstinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *TCC*, 2015.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, 2013.

[GJPS08] Vipul Goyal, Abhishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute based encryption. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, 2008.

[GKW17a] Rishab Goyal, Venkata Koppula, and Brent Waters. Separating IND-CPA and circular security for unbounded length key cycles. In *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I*, 2017.

[GKW17b] Rishab Goyal, Venkata Koppula, and Brent Waters. Separating semantic and circular security for symmetric-key bit encryption from the learning with errors assumption. In *EUROCRYPT*, 2017.

[GLSW15] Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 151–170, 2015.

[GLW14] Craig Gentry, Allison B. Lewko, and Brent Waters. Witness encryption from instance independent assumptions. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 426–443, 2014.

[GMM+16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. In *Theory of Cryptography Conference*, pages 241–268. Springer, 2016.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, 2006.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, 2013.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from lwe. In *Annual Cryptology Conference*, 2015.

[Hal15] Shai Halevi. Graded encoding, variations on a scheme. Cryptology ePrint Archive, Report 2015/866, 2015.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[HJ16] Yupu Hu and Huiwen Jia. Cryptanalysis of ggh map. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2016.

[KRW15] Venkata Koppula, Kim Ramchen, and Brent Waters. Separations in circular security for arbitrary length key cycles. In *Theory of Cryptography Conference (TCC)*, 2015.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*, EUROCRYPT'08, 2008.

[KW16] Venkata Koppula and Brent Waters. Circular security counterexamples for arbitrary length cycles from LWE. In *CRYPTO*, 2016.

[Lau02] Peeter Laud. Encryption cycles and two views of cryptography. In *NORDSEC 2002 - Proceedings of the 7th Nordic Workshop on Secure IT Systems (Karlstad University Studies 2002:31*, pages 85–100, 2002.

[Lin16a] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2016.

[Lin16b] Huijia Lin. Indistinguishability obfuscation from ddh on 5-linear maps and locality-5 prgs. Cryptology ePrint Archive, Report 2016/1096, 2016.

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, 2016.

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, pages 455–479, 2010.

[LW11] Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, pages 568–588, 2011.

[MO14] Antonio Marcedone and Claudio Orlandi. Obfuscation ⇒ (IND-CPA security !⇒ circular security). In *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, pages 77–90, 2014.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 700–718, 2012.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 21–39, 2013.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, April 2007.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over ggh13. In *Annual Cryptology Conference*, 2016.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 333–342, 2009.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *International Cryptology Conference*, 2014.

[RAD78] Ron Rivest, Leonard Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–180, 1978.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93, 2005.

[Rot13] Ron Rothblum. On the circular security of bit-encryption. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 579–598, 2013.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484, 2014.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *CRYPTO*, pages 619–636, 2009.

[Wat12] Brent Waters. Functional encryption for regular languages. In *CRYPTO*, 2012.

[WFS03] Brent R Waters, Edward W Felten, and Amit Sahai. Receiver anonymity via incomparable public keys. In *Proceedings of the 10th ACM conference on Computer and communications security*, 2003.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under lwe. Cryptology ePrint Archive, Report 2017/276, 2017.

# A Background: Attribute-Based Encryption and Predicate Encryption

## A.1 Key-Policy Attribute Based Encryption

A key-policy attribute based encryption (KP-ABE) scheme $\mathcal{ABE}$, for set of attribute spaces $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$, predicate classes $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ and message spaces $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$, consists of four polytime algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, \mathcal{X}_\lambda, \mathcal{C}_\lambda, \mathcal{M}_\lambda) \to (\mathsf{pp}, \mathsf{msk})$. The setup algorithm takes as input the security parameter $\lambda$ and a description of attribute space $\mathcal{X}_\lambda$, predicate class $\mathcal{C}_\lambda$ and message space $\mathcal{M}_\lambda$, and outputs the public parameters $\mathsf{pp}$ and the master secret key $\mathsf{msk}$.
- $\mathsf{Enc}(\mathsf{pp}, x, m) \to \mathsf{ct}$. The encryption algorithm takes as input public parameters $\mathsf{pp}$, an attribute $x \in \mathcal{X}_\lambda$ and a message $m \in \mathcal{M}_\lambda$. It outputs a ciphertext $\mathsf{ct}$.
- $\mathsf{KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$. The key generation algorithm takes as input master secret key $\mathsf{msk}$ and a predicate $C \in \mathcal{C}_\lambda$. It outputs a secret key $\mathsf{sk}_C$.
- $\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}) \to m$ or $\perp$. The decryption algorithm takes as input a secret key $\mathsf{sk}_C$ and a ciphertext $\mathsf{ct}$. It outputs either a message $m \in \mathcal{M}_\lambda$ or a special symbol $\perp$.

**Correctness.** A key-policy attribute based encryption scheme is said to be correct if for all $\lambda \in \mathbb{N}$, $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{X}_\lambda, \mathcal{C}_\lambda, \mathcal{M}_\lambda)$, for all $x \in \mathcal{X}_\lambda$, $C \in \mathcal{C}_\lambda$, $m \in \mathcal{M}_\lambda$, $\mathsf{sk}_C \leftarrow \mathsf{KeyGen}(\mathsf{msk}, C)$, $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, x, m)$, the following holds

$$C(x) = 1 \Rightarrow \Pr\left[\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}) = m\right] \geq 1 - \mathrm{negl}_1(\lambda),$$
$$C(x) = 0 \Rightarrow \Pr\left[\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}) = \perp\right] \geq 1 - \mathrm{negl}_2(\lambda),$$

where $\mathrm{negl}_1(\cdot), \mathrm{negl}_2(\cdot)$ are negligible functions, and the probabilities are taken over the random coins used during key generation and encryption procedures.

**Security.** The standard notion of security for a KP-ABE scheme is that of full or adaptive security. It is formally defined as follows.

**Definition A.1.** A key-policy attribute based encryption scheme $\mathcal{ABE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is said to be fully secure if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$, such that the following holds:

$$\left| \Pr\left[ \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c} (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{X}_\lambda, \mathcal{C}_\lambda, \mathcal{M}_\lambda) \\ ((m_0, m_1), x) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{pp}) \\ b \leftarrow \{0,1\};\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, x, m_b) \end{array} \right] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda)$$

where every predicate query $C$, made by adversary $\mathcal{A}$ to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle, must satisfy the condition that $C(x) = 0$.

## A.2 Predicate Encryption

A predicate encryption (PE) scheme $\mathcal{PE}$, for set of attribute spaces $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$, predicate classes $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ and message spaces $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$, consists of four polytime algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$. They have the same syntax and correctness condition as that of a KP-ABE scheme.

The standard notion of security for a PE scheme is that of full attribute hiding (or 2-sided security). A weaker notion of security is of 1-sided security which is captured by the following indistinguishability based definition.

**Definition A.2** (1-Sided Security)**.** A predicate encryption scheme $\mathcal{PE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is said to be 1-sided secure if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$, such that the following holds:

$$\left| \Pr \left[ \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c} (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{X}_\lambda, \mathcal{C}_\lambda, \mathcal{M}_\lambda) \\ ((m_0, x_0), (m_1, x_1)) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{pp}) \\ b \leftarrow \{0,1\};\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, x_b, m_b) \end{array} \right] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda)$$

where every predicate query $C$, made by adversary $\mathcal{A}$ to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle, must satisfy the condition that $C(x_0) = C(x_1) = 0$.

The notion of 1-sided security for PE schemes could alternatively be captured by a simulation based definition. The following definition was used by Gorbunov, Vaikuntanathan and Wee [GVW15]. Later in Appendix F, we show that if a PE scheme satisfies Definition A.2, then it also satisfies Definition A.3.

**Definition A.3** (SIM-1-Sided Security)**.** A predicate encryption scheme $\mathcal{PE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is said to be SIM-1-sided secure if there exists a simulator $\mathsf{Sim}$ such that for every stateful PPT adversary $\mathcal{A}$, the following holds:

$$\left\{ \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}) : \begin{array}{c} (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{X}_\lambda, \mathcal{C}_\lambda, \mathcal{M}_\lambda) \\ (m, x) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{pp}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, x, m) \end{array} \right\}$$

$$\approx_c$$

$$\left\{ \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}) : \begin{array}{c} (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{X}_\lambda, \mathcal{C}_\lambda, \mathcal{M}_\lambda) \\ (m, x) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{pp}) \\ \mathsf{ct} \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{pp}, 1^{|x|}, 1^{|m|}) \end{array} \right\}$$

where every predicate query $C$, made by adversary $\mathcal{A}$ to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle, must satisfy the condition that $C(x) = 0$.

# B Background: Cycle Testers

## B.1 Public Key $k$-Cycle Tester

In a recent work, Bishop, Hohenberger and Waters [BHW15] introduced a generic framework for creating circular security counterexamples. In this *cycle tester* framework, there are four algorithms - Setup, KeyGen, Encrypt and *Test*. The setup algorithm takes as input the security parameter $\lambda$ and cycle length parameter $k$ outputs the public parameters. The key generation algorithm uses the public parameters to output a public-secret key pair. The encryption algorithm takes a public key and message as input, and outputs a ciphertext. Finally, the testing algorithm takes as input $k$ public keys and $k$ ciphertexts, and outputs 1 if the $k$ ciphertexts form a key cycle, else it outputs 0. *Note that in this framework, there is no decryption algorithm.* The security requirement is identical to the IND-CPA security game. The following description is taken from [BHW15].

**Definition B.1** ($k$-Cycle Tester)**.** A *cycle tester* $\Gamma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Test})$ for message space $\mathcal{M}$ and secret key space $\{0,1\}^s$ is a tuple of algorithms (where $s = s(\lambda)$) specified as follows:

- $\mathsf{Setup}(1^\lambda, 1^k) \rightarrow \mathsf{pp}$. The setup algorithm takes as input the security parameter $\lambda$ and the length of cycle $k$. It outputs the public parameters $\mathsf{pp}$.
- $\mathsf{KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{pk}, \mathsf{sk})$. The key generation algorithm takes as input the public parameters $\mathsf{pp}$ and outputs a public key $\mathsf{pk}$ and secret key $\mathsf{sk} \in \{0,1\}^s$.

- $\mathsf{Enc}(\mathsf{pk}, m \in \mathcal{M}) \to \mathsf{ct}$. The encryption algorithm takes as input a public key $\mathsf{pk}$ and a message $m \in \mathcal{M}$ and outputs a ciphertext $\mathsf{ct}$.
- $\mathsf{Test}(\mathbf{pk}, \mathbf{ct}) \to \{0,1\}$. On input $\mathbf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_k)$ and $\mathbf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, the testing algorithm outputs a bit in $\{0,1\}$.

The algorithms must satisfy the following properties.

1. (Testing Correctness) There exists a polynomial $p(\cdot)$ such that for all security parameters $\lambda$, for all cycle lengths $k$, the Test algorithm's advantage (given by the following expression) is at least $1/p(\lambda)$.

$$\left| \Pr\left[1 \leftarrow \mathsf{Test}(\mathbf{pk}, \mathbf{ct}) \ : \ \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^k); (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{sk}_{(i \bmod k)+1}) \end{array}\right] - \Pr\left[1 \leftarrow \mathsf{Test}(\mathbf{pk}, \mathbf{ct}) \ : \ \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^k); (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, 0^s) \end{array}\right] \right|$$

2. ($\mathsf{IND\text{-}CPA}$ Security) Let $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \cdot)$ be an encryption scheme with empty decryption algorithm. The scheme $\Pi$ must satisfy the $\mathsf{IND\text{-}CPA}$ security definition (Definition 2.4).

## B.2 Public Key Bit Encryption 1-Cycle Tester

The BHW framework does not directly work for generating circular security separations for bit-encryption. In a recent work, Goyal, Koppula and Waters [GKW17b] provide a *bit-encryption cycle tester framework* for symmetric-key bit encryption along the lines of BHW framework. Below we extend their framework in the public key setting.

**Definition B.2.** (Bit Encryption 1-Cycle Tester) A cycle tester $\Gamma = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Test})$ for message space $\{0,1\}$ and secret key space $\{0,1\}^s$ is a tuple of algorithms (where $s = s(\lambda)$) specified as follows:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$. The setup algorithm takes as input the security parameter $\lambda$, and outputs a public-secret key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk}, m \in \{0,1\}) \to \mathsf{ct}$. The encryption algorithm takes as input a public key $\mathsf{pk}$ and a message $m \in \{0,1\}$, and outputs a ciphertext $\mathsf{ct}$.
- $\mathsf{Test}(\mathsf{pk}, \mathbf{ct}) \to \{0,1\}$. The testing algorithm takes as input a public key $\mathsf{pk}$ and a sequence of $s$ ciphertexts $\mathbf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_s)$, and outputs a bit in $\{0,1\}$.

The algorithms must satisfy the following properties.

1. (Testing Correctness) There exists a polynomial $p(\cdot)$ such that for all security parameters $\lambda$, the Test algorithm's advantage (given by the following expression) is at least $1/p(\lambda)$.

$$\left| \Pr\left[1 \leftarrow \mathsf{Test}(\mathsf{pk}, \mathbf{ct}) \ : \ \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{sk}_i) \end{array}\right] - \Pr\left[1 \leftarrow \mathsf{Test}(\mathsf{pk}, \mathbf{ct}) \ : \ \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, 0) \end{array}\right] \right|$$

2. ($\mathsf{IND\text{-}CPA}$ Security) Let $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \cdot)$ be an encryption scheme with empty decryption algorithm. The scheme $\Pi$ must satisfy the $\mathsf{IND\text{-}CPA}$ security definition (Definition 2.4).

### B.3 Public Key Cycle Tester for Unbounded Length Cycles

The BHW framework does not give circular security separations for unbounded length key cycles since the $\mathsf{Setup}$ algorithm in their framework takes as input the length of cycle $k$. Below we extend the BHW framework for unbounded length key cycles.

**Definition B.3** (Unbounded Length Cycle Tester)**.** A *cycle tester* $\Gamma = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Test})$ for message space $\mathcal{M}$ and secret key space $\{0,1\}^s$ is a tuple of algorithms (where $s = s(\lambda)$) specified as follows:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$. The setup algorithm takes as input the security parameter $\lambda$ and outputs a public key $\mathsf{pk}$ and secret key $\mathsf{sk} \in \{0,1\}^s$.
- $\mathsf{Enc}(\mathsf{pk}, m \in \mathcal{M}) \to \mathsf{ct}$. The encryption algorithm takes as input a public key $\mathsf{pk}$ and a message $m \in \mathcal{M}$ and outputs a ciphertext $\mathsf{ct}$.
- $\mathsf{Test}(1^k, \mathbf{pk}, \mathbf{ct}) \to \{0,1\}$. On input the key cycle length $k$, public keys $\mathbf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_k)$ and secret keys $\mathbf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, the testing algorithm outputs a bit in $\{0,1\}$.

The algorithms must satisfy the following properties.

1. (Testing Correctness) There exists a polynomial $p(\cdot)$ such that for all security parameters $\lambda$, all cycle lengths $k$, the Test algorithm's advantage (given by the following expression) is at least $1/p(\lambda)$.

$$\left| \Pr\left[1 \leftarrow \mathsf{Test}(1^k, \mathbf{pk}, \mathbf{ct}) \ : \ \begin{array}{l}(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{sk}_{(i \bmod k)+1})\end{array}\right] - \Pr\left[1 \leftarrow \mathsf{Test}(1^k, \mathbf{pk}, \mathbf{ct}) \ : \ \begin{array}{l}(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, 0^s)\end{array}\right] \right|$$

2. ($\mathsf{IND\text{-}CPA}$ Security) Let $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \cdot)$ be an encryption scheme with empty decryption algorithm. The scheme $\Pi$ must satisfy the $\mathsf{IND\text{-}CPA}$ security definition (Definition 2.4).

# C Lockable Obfuscators: Extending the Message Space

In this section, we show that a lockable obfuscation scheme with message space $\{0,1\}$ can be extended to a lockable obfuscation scheme with message space $\{0,1\}^\ell$ for any polynomial $\ell$.

Let $(\mathsf{Obf}, \mathsf{Eval})$ be a lockable obfuscation scheme with message space $\{0,1\}$. We will also require the following tools for this transformation:

- A pairwise independent hash function family $\mathcal{H}$ with domain $\{0,1\}^{\ell_{\text{out}}}$ and co-domain $\{0,1\}^{\ell_{\mathcal{H}}}$
- A pseudorandom generator $\text{PRG} : \{0,1\}^{\ell_{\mathcal{H}}} \to \{0,1\}^{\ell \cdot \lambda}$

Consider the following lockable obfuscation scheme $(\mathsf{Obf}', \mathsf{Eval}')$ for circuit class $\mathcal{C}_{\ell_{\text{in}}, \ell_{\text{out}}, d}$ and message space $\{0,1\}^\ell$.

- $\mathsf{Obf}'(1^\lambda, P, \mathsf{msg}, \alpha)$: Let $\mathsf{msg}_i$ denote the $i^{th}$ bit of $\mathsf{msg}$ for $i \le \ell$, and $s_{[i,j]}$ be substring of $s$ consisting starting with the $i^{th}$ bit of $s$ and ending with the $j^{th}$ bit.

  The obfuscation algorithm first chooses a hash function $h$. Let $Q^i$ be a circuit that takes as input $x \in \{0,1\}^{\ell_{\text{in}}}$ and outputs $\text{PRG}(h(P(x)))_{[(i-1)\cdot\lambda+1, i\cdot\lambda]}$.

  The obfuscation algorithm computes $\widetilde{P}_i \leftarrow \mathsf{Obf}(1^\lambda, Q^i, \mathsf{msg}_i, \text{PRG}(h(\alpha))_{[(i-1)\cdot\lambda+1, i\cdot\lambda]})$ for each $i \le \ell$. The final obfuscated program is $\left(\widetilde{P}_1, \ldots, \widetilde{P}_\ell\right)$.

- $\mathsf{Eval}'\left(\left(\widetilde{P}_1, \ldots, \widetilde{P}_\ell\right), x\right)$: The evaluation algorithm computes $y_i = \mathsf{Eval}(\widetilde{P}_i, x)$ for each $i \le \ell$. If any $y_i = \bot$, it outputs $\bot$. Else, it outputs $(y_1, \ldots, y_\ell)$.

**Correctness.** We will show that the above scheme satisfies the semi-statistical correctness definition if the underlying scheme also satisfies the semi-statistical correctness notion. Fix any security parameter $\lambda$, program $P$, string $\alpha$, input $x$ and message $\mathsf{msg}$. If $P(x) = \alpha$, then it follows directly that $\mathsf{Eval}'\left(\mathsf{Obf}'\left(1^\lambda, P, \mathsf{msg}, \alpha\right), x\right) = \mathsf{msg}$.

If $P(x) \neq \alpha$, then we need to show that with high probability (over the coins of obfuscation), the evaluation algorithm outputs $\perp$. First, observe that the evaluation algorithm $\mathsf{Eval}'$ outputs $\perp$ whenever $y_i = \perp$ for any $i$ (i.e., any of $\ell$ $\mathsf{Eval}$ invocations outputs $\perp$). Thus, we only need to argue that the probability all $\ell$ $\mathsf{Eval}$ operations output some string $y \neq \perp$ is negligible.

Let $\mathsf{Bad}$ denote the event that $\mathrm{PRG}(h(P(x))) = \mathrm{PRG}(h(\alpha))$, and let $\mathcal{E}$ be the event that $\mathsf{Eval}'((\widetilde{P}_1, \ldots, \widetilde{P}_\ell), x) \neq \perp$. First, by a similar argument to that used in proof of Claim 4.3, we can argue that $\Pr[\mathsf{Bad}] \leq 2^{-\ell_{\mathcal{H}}}$. Next, we know that if $\mathsf{Bad}$ does not occur, then there exists an $i$ such that $\mathrm{PRG}(h(P(x)))_{[(i-1)\cdot\lambda+1, i\cdot\lambda]} \neq \mathrm{PRG}(h(\alpha))_{[(i-1)\cdot\lambda+1, i\cdot\lambda]}$ (i.e., the $i^{th}$ segment in the $\mathrm{PRG}(h(P(x)))$ does not match the corresponding lock value). Finally, using the underlying obfuscator's semi-statistical correctness, we get that $\Pr[\mathcal{E} \mid \overline{\mathsf{Bad}}] \leq \mathrm{negl}(\lambda)$. Hence with all but negligible probability, the evaluation algorithm outputs $\perp$ as $\Pr[\mathcal{E}] \leq \Pr[\mathcal{E} \mid \overline{\mathsf{Bad}}] + \Pr[\mathsf{Bad}] \leq \mathrm{negl}(\lambda)$. This concludes the proof of correctness.

**Security.** Security follows from a hybrid argument. The simulator will run the underlying scheme's simulator $\ell$ times. To prove that this is indistinguishable from the honestly computed obfuscation, we define $\ell + 2$ intermediate hybrid obfuscations. In the first hybrid, we switch $h(\alpha)$ to be a uniformly random string. This follows from the Leftover Hash Lemma. Next, we switch $\mathrm{PRG}(h(\alpha))$ to a uniformly random $\ell \cdot \lambda$ bit string. This follows from the security of PRG. At this point, each of the locks for each of the $\ell$ calls to $\mathsf{Obf}$ are independent and uniformly random strings. We can now simulate each of them independently. In the $(i+2)^{th}$ one, the challenger simulates the first $i$ obfuscations, and honestly computes the remaining $\ell - i$ obfuscations. The $(i+1)^{th}$ and $(i+2)^{th}$ hybrids are computationally indistinguishable due to security of the underlying obfuscation scheme.

# D Lockable Obfuscation with Statistical Correctness

In this section, we show how to construct lockable obfuscation with *statistical correctness*. The construction and proofs will be very similar to the corresponding parts from Section 4. In order to achieve this stronger notion of correctness, we will require low depth *injective* pseudorandom generators. Currently, we do not know how to instantiate such PRGs from the LWE assumption.

For any polynomials $\ell_{\mathrm{in}}, \ell_{\mathrm{out}}, d$ such that $\ell_{\mathrm{out}} = \Omega(\lambda)$, we construct an lockable obfuscation scheme $(\mathsf{Obf}, \mathsf{Eval})$ with statistical correctness for the circuit class $\mathcal{C}_{\ell_{\mathrm{in}}, \ell_{\mathrm{out}}, d}$. The messge space for our construction will be $\{0,1\}$. The tools required for this construction are as follows:

- A compact leveled homomorphic bit encryption scheme $(\mathsf{LHE.Setup}, \mathsf{LHE.Enc}, \mathsf{LHE.Eval}, \mathsf{LHE.Dec})$ with decryption circuit of depth $d_{\mathsf{Dec}}(\lambda)$ and ciphertexts of length $\ell_{\mathsf{ct}}(\lambda)$.
- An injective pseudorandom generator $\mathrm{PRG} : \{0,1\}^{\ell_{\mathrm{out}}(\lambda)} \to \{0,1\}^{\ell_{\mathrm{PRG}}(\lambda)}$ of depth $d_{\mathrm{PRG}}(\lambda)$.

Fix any $\epsilon < 1/2$. Let $\chi$ be a $B$-bounded discrete Gaussian distribution with parameter $\sigma$ such that $B = \sqrt{m} \cdot \sigma$. Let $n, m, \ell, \sigma, q, \mathsf{Bd}$ be parameters with the following constraints:

- $n = \mathsf{poly}(\lambda)$ and $q \leq 2^{n^\epsilon}$ (for LWE security)
- $\ell_{\mathrm{out}} = \omega(\log \lambda)$ (for PRG security)
- $m = \Omega(n \cdot \log q)$ (for $\mathsf{SamplePre}$)
- $\sigma = \omega(\sqrt{n \cdot \log q \cdot \log m})$ (for Preimage Well Distributedness)
- $\ell_{\mathrm{PRG}} = n \cdot m \cdot \log q + \omega(\log n)$ (for applying Leftover Hash Lemma)
- $\mathsf{Bd} = \ell_{\mathrm{PRG}} \cdot L \cdot (m^2 \cdot \sigma)^L < q^{1/4}$ (for correctness of scheme)
  (where $L = \ell_{\mathrm{out}} \cdot \ell_{\mathsf{ct}} \cdot 4^{d_{\mathsf{Dec}} + d_{\mathrm{PRG}}}$)

As before, it is important that $L = \lambda^c$ for some constant $c$ and $\mathsf{Bd} = \ell_{\mathrm{PRG}} \cdot L \cdot (m^2 \cdot \sigma)^L < q^{1/4}$. The constant $c$ depends on the LHE scheme and PRG.

We will now describe the obfuscation and evaluation algorithms.

- $\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha)$: The obfuscation algorithm takes as input a program $P \in \mathcal{C}_{\ell_{\mathrm{in}},\ell_{\mathrm{out}},d}$, message $\mathsf{msg} \in \{0,1\}$ and $\alpha \in \{0,1\}^{\ell_{\mathrm{out}}}$. The obfuscator proceeds as follows:

  1. First, it chooses the LHE key pair as $(\mathsf{lhe.sk}, \mathsf{lhe.ek}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^{d \log d})$.
  2. Next, it encrypts the program $P$. It sets $\mathbf{ct} \leftarrow \mathsf{LHE.Enc}(\mathsf{lhe.sk}, P)$.
  3. Let $\beta = \mathrm{PRG}(\alpha)$.
  4. Next, consider the following circuit $Q$ which takes as input $\ell_{\mathrm{out}} \cdot \ell_{\mathsf{ct}}$ bits of input and outputs $\ell_{\mathrm{PRG}}$ bits. $Q$ takes as input $\ell_{\mathrm{out}}$ LHE ciphertexts $\{\mathsf{ct}_i\}_{i \le \ell_{\mathrm{out}}}$, has LHE secret key $\mathsf{lhe.sk}$ hardwired and computes the following — (1) it decrypts each input ciphertext $\mathsf{ct}_i$ (in parallel) to get string $x$ of length $\ell_{\mathrm{out}}$ bits, (2) it applies the PRG on $x$ and outputs $\mathrm{PRG}(x)$. Concretely, $Q(\mathsf{ct}_1, \ldots, \mathsf{ct}_{\ell_{\mathrm{out}}}) = \mathrm{PRG}\left(\mathsf{LHE.Dec}(\mathsf{lhe.sk}, \mathsf{ct}_1) \,||\, \cdots \,||\, \mathsf{LHE.Dec}(\mathsf{lhe.sk}, \mathsf{ct}_{\ell_{\mathrm{out}}})\right)$.

     For $i \le \ell_{\mathrm{PRG}}$, we use $\mathsf{BP}^{(i)}$ to denote the fixed-input selector permutation branching program that outputs the $i^{th}$ bit of output of circuit $Q$. Note that $Q$ has depth at most $d_{\mathsf{tot}} = d_{\mathsf{Dec}} + d_{\mathrm{PRG}}$. By Corollary 2.2, we know that each branching program $\mathsf{BP}^{(i)}$ has length $L = \ell_{\mathrm{out}} \cdot \ell_{\mathsf{ct}} \cdot 4^{d_{\mathsf{tot}}}$ and width 5.
  5. Finally, the obfuscator creates matrix components which enable the evaluator to compute $\mathsf{msg}$ if it has an input strings (ciphertexts) $\mathsf{ct}_1, \ldots, \mathsf{ct}_{\ell_{\mathrm{out}}}$ such that $Q(\mathsf{ct}_1, \ldots, \mathsf{ct}_{\ell_{\mathrm{out}}}) = \beta$. Concretely, it runs the (randomized) routine $\mathsf{Comp\text{-}Gen}$ (defined in Figure 1). This routine takes as input the circuit $Q$ in the form of $\ell_{\mathrm{PRG}}$ branching programs $\{\mathsf{BP}^{(i)}\}_i$, string $\beta$ and message $\mathsf{msg}$. Let $\left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, \left\{\mathbf{C}_j^{(i,0)}, \mathbf{C}_j^{(i,1)}\right\}_{i,j}\right) \leftarrow \mathsf{Comp\text{-}Gen}(\{\mathsf{BP}^{(i)}\}_i, \beta, \mathsf{msg})$.
  6. The final obfuscated program consists of the LHE evaluation key $\mathsf{ek} = \mathsf{lhe.ek}$, LHE ciphertexts $\mathbf{ct}$, together with the components $\left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, \left\{(\mathbf{C}_j^{(i,0)}, \mathbf{C}_j^{(i,1)})\right\}_{i,j}\right)$.

- $\mathsf{Eval}(\tilde{P}, x)$: The evaluation algorithm takes as input $\tilde{P} = \left(\mathsf{ek}, \mathbf{ct}, \left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, \left\{(\mathbf{C}_j^{(i,0)}, \mathbf{C}_j^{(i,1)})\right\}_{i,j}\right)$ and input $x \in \{0,1\}^{\ell_{\mathrm{in}}}$. It performs the following steps.

  1. The evaluator first constructs a universal circuit $U_x(\cdot)$ with $x$ hardwired as input. This universal circuit takes a circuit $C$ as input and outputs $U_x(C) = C(x)$. Using the universal circuit of Cook and Hoover [CH85], it follows that $U_x(\cdot)$ has depth $O(d)$.
  2. Next, it performs homomorphic evaluation on $\mathbf{ct}$ using circuit $U_x(\cdot)$. It computes $\widetilde{\mathbf{ct}} = \mathsf{LHE.Eval}(\mathsf{ek}, U_x(\cdot), \mathbf{ct})$. Note that $\ell_{\mathsf{ct}} \cdot \ell_{\mathrm{out}}$ denotes the length of $\widetilde{\mathbf{ct}}$ (as a bitstring), and let $\widetilde{\mathbf{ct}}_i$ denote the $i^{th}$ bit of $\widetilde{\mathbf{ct}}$.
  3. The evaluator then obliviously evaluates the $\ell_{\mathrm{PRG}}$ branching programs on input $\widetilde{\mathbf{ct}}$ using the matrix components. It calls the component evaluation algorithm $\mathsf{Comp\text{-}Eval}$ (defined in Figure 2). Let $y = \mathsf{Comp\text{-}Eval}\left(\widetilde{\mathbf{ct}}, \left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, \left\{(\mathbf{C}_j^{(i,0)}, \mathbf{C}_j^{(i,1)})\right\}_{i,j}\right)\right)$. The evaluator outputs $y$.

## D.1 Correctness

We will prove that the lockable obfuscation scheme described above satisfies the statistical correctness property (see Definition 3.2). First, let us recall the following lemma (a weaker version of this lemma was proven in Section 4.1).

**Lemma D.1.** For any set of branching programs $\{\mathsf{BP}^{(i)}\}_{i \le \ell_{\mathrm{PRG}}}$, string $\beta \in \{0,1\}^{\ell_{\mathrm{PRG}}}$, message $\mathsf{msg} \in \{0,1\}$ and input $z$,

1. if $\mathsf{BP}^{(i)}(z) = \beta_i$ for all $i \leq \ell_{\mathrm{PRG}}$, then $\mathsf{Comp\text{-}Eval}(z, \mathsf{Comp\text{-}Gen}(\{\mathsf{BP}^{(i)}\}_i, \beta, \mathsf{msg})) = \mathsf{msg}$.
2. if $\mathsf{BP}^{(i)}(z) \neq \beta_i$ for some $i \leq \ell_{\mathrm{PRG}}$, then

$$\Pr[\mathsf{Comp\text{-}Eval}(z, \mathsf{Comp\text{-}Gen}(\{\mathsf{BP}^{(i)}\}_i, \beta, \mathsf{msg})) = \bot] \geq 1 - \mathrm{negl}(\lambda)/2^{\ell_{\mathrm{in}}}.$$

This lemma is identical to Lemma 4.1, except that if $\mathsf{BP}^{(i)}(z) \neq \beta_i$ for some $i \leq \ell_{\mathrm{PRG}}$, then the probability of error needs to be bounded by $\mathrm{negl}(\lambda)/2^{\ell_{\mathrm{in}}}$ instead of just $\mathrm{negl}(\lambda)$. This is required for our union bound argument. The proof of this lemma is same as that of Lemma 4.1, and the final bound of $\mathrm{negl}(\lambda)/2^{\ell_{\mathrm{in}}}$ holds if we ensure that $(\mathsf{Bd}/q)^{(m-n)\cdot n} < 2^{-2\ell_{\mathrm{in}}}$.

Using the above lemma, we can now argue the correctness of our scheme. First, we need to show correctness for the case when $P(x) = \alpha$.

**Claim D.1.** For all security parameters $\lambda$, inputs $x \in \{0,1\}^{\ell_{\mathrm{in}}}$, programs $P \in \mathcal{C}_{\ell_{\mathrm{in}},\ell_{\mathrm{out}},d}$ and messages $\mathsf{msg} \in \{0,1\}$, if $P(x) = \alpha$, then

$$\mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha)), x) = \mathsf{msg}.$$

*Proof.* First, the obfuscator encrypts the program $P$ using an LHE secret key $\mathsf{lhe.sk}$, and sets $\mathsf{ct} \leftarrow \mathsf{LHE.Enc}(\mathsf{lhe.sk}, P)$. The evaluator evaluates the LHE ciphertext on universal circuit $U_x(\cdot)$, which results in an evaluated ciphertext $\widetilde{\mathbf{ct}}$. Now, by the correctness of the LHE scheme, decryption of $\widetilde{\mathbf{ct}}$ using $\mathsf{lhe.sk}$ outputs $\alpha$. Therefore, $\mathrm{PRG}(\mathsf{LHE.Dec}(\mathsf{lhe.sk}, \widetilde{\mathbf{ct}})) = \beta$. Then, using Lemma 4.1, we can argue that $\mathsf{Comp\text{-}Eval}$ outputs $\mathsf{msg}$, and thus $\mathsf{Eval}$ outputs $\mathsf{msg}$. ∎

**Claim D.2.** For all security parameters $\lambda$, programs $P \in \mathcal{C}_{n,m,d}(\lambda)$, $\alpha \in \{0,1\}^{m(\lambda)}$ and $\mathsf{msg} \in \mathcal{M}$,

$$\Pr[\exists\ x \ \text{ s.t. } P(x) \neq \alpha \text{ and } \mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha), x) = \mathsf{msg}] \leq \mathrm{negl}(\lambda)$$

where the probability is taken over the random coins used during obfuscation.

*Proof.* Fix any security parameter $\lambda$, program $P$, $\alpha$, $x$ such that $P(x) \neq \alpha$ and message $\mathsf{msg}$. Let us now consider the events

$$\mathsf{Error} : \exists\ x \text{ s.t. } P(x) \neq \alpha \text{ and } \mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha), x) \neq \bot$$

$$\mathsf{Error}_x : \mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha), x) \neq \bot$$

Clearly, $\Pr[\mathsf{Error}] \leq \sum_{x:P(x)\neq\alpha} \Pr[\mathsf{Error}_x]$. Therefore, it suffices to show that $\Pr[\mathsf{Error}_x]$ is at most $\mathrm{negl}(\lambda)/2^{\ell_{\mathrm{in}}}$.

First, note that for all $x \in \{0,1\}^{\ell_{\mathrm{in}}}$, if $P(x) \neq \alpha$, then $\mathrm{PRG}(P(x)) \neq \mathrm{PRG}(\alpha)$. This follows from the injectiveness of PRG. Fix any input $x$ such that $P(x) \neq \alpha$, LHE keys $(\mathsf{lhe.sk}, \mathsf{lhe.ek})$ and ciphertext $\mathbf{ct} \leftarrow \mathsf{LHE.Enc}(\mathsf{lhe.sk}, P)$. Let $\widetilde{\mathbf{ct}} = \mathsf{LHE.Eval}(\mathsf{ek}, U_x(\cdot), \mathsf{ct})$ and $Q$ be the program that, on input a LHE ciphertext, first decrypts it using $\mathsf{sk}$, then computes $\mathrm{PRG}(y)$ (where $y$ is the decrypted string). Let $\mathsf{BP}^{(i)}$ denote the branching program computing the $i^{th}$ output bit of $Q$. Using the correctness of LHE decryption, it follows that $Q(\widetilde{\mathbf{ct}}) = \mathrm{PRG}(P(x)) \neq \beta$,[21] and therefore there exists some $i \leq \ell_{\mathrm{PRG}}$ such that $\mathsf{BP}^{(i)}(\widetilde{\mathbf{ct}}) \neq \beta_i$. Now, $\Pr[\mathsf{Error}_x] = \Pr[\mathsf{Comp\text{-}Eval}\left(\widetilde{\mathbf{ct}}, \mathsf{Comp\text{-}Gen}\left(\{\mathsf{BP}^{(i)}\}_i, \beta, \mathsf{msg}\right)\right)]$. Using Lemma D.1, we can conclude that for all $\mathsf{msg} \in \{0,1\}$, this probability is at most $\mathrm{negl}(\lambda)/2^{\ell_{\mathrm{in}}}$. Hence, $\Pr[\mathsf{Error}] \leq \mathrm{negl}(\lambda)$.

∎

## D.2 Security

We will now prove the scheme secure as per Definition 3.4. This proof is very similar to the one in Section 4.2. The only difference here is that we do not require a pairwise independent hash function.

[21] Recall $\beta = \mathrm{PRG}(\alpha)$

### D.2.1 Simulator Sim

The simulator first chooses the parameters $n, m, q, \sigma, \ell_{\mathrm{PRG}}$ as in the original scheme. Next, it chooses LHE secret/evaluation keys. It computes $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^{d \log d})$ (note that the depth $d$ of the circuit class is fixed for the scheme). It then computes an encryption of $\mathbf{0}^{|P|}$. Let $\mathsf{ct} \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}, \mathbf{0}^{|P|})$. Finally, the simulator chooses $\ell_{\mathrm{PRG}}$ matrices $\mathbf{B}_{0,1}^{(i)} \leftarrow \mathbb{Z}_q^{n \times m}$ and low norm matrices $\{(\mathbf{C}_j^{(i,0)}, \mathbf{C}_j^{(i,1)})\}_{i,j}$ for $i \leq \ell_{\mathrm{PRG}}$, $j \leq L$ where $\mathbf{C}_j^{(i,b)} \leftarrow \chi^{m \times m}$. The obfuscation consists of the LHE evaluation key $\mathsf{ek}$, ciphertext $\mathsf{ct}$, together with the components $\left( \left\{ \mathbf{B}_{0,1}^{(i)} \right\}_i, \left\{ (\mathbf{C}_j^{(i,0)}, \mathbf{C}_j^{(i,1)}) \right\}_{i,j} \right)$.

To prove security, we will define a sequence of hybrids. The proofs showing indistinguishability of hybrid experiments are identical to the ones in Section 4.2.

### D.2.2 Sequence of Hybrid Games

Fix any program $P$, message $\mathsf{msg} \in \{0,1\}$.

Game 0**:** This corresponds to the original security game.

1. The challenger first chooses the LWE parameters $n$, $m$, $q$, $\sigma$, $\chi$ and $\ell_{\mathrm{PRG}}$. Recall $L$ denotes the length of the branching programs.
2. It chooses $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^{d \log d})$ and sets $\mathsf{ct} \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}, P)$.
3. Next, it chooses a uniformly random string $\alpha \leftarrow \{0,1\}^{\ell_{\mathrm{out}}}$ and sets $\beta = \mathrm{PRG}(\alpha)$.
4. Next, consider the following program $Q$. It takes as input an LHE ciphertext $\mathsf{ct}$, has $\mathsf{sk}$ hardwired and does the following: it decrypts the input ciphertext $\mathsf{ct}$ to get string $y$ and outputs $\mathrm{PRG}(y)$. For $i \leq \ell_{\mathrm{PRG}}(\lambda)$, let $\mathsf{BP}^{(i)}$ denote the branching program that outputs the $i^{th}$ bit of $\mathrm{PRG}(y)$.
5. It chooses $\ell_{\mathrm{PRG}}$ uniformly random matrices $\mathbf{B}_L^{(i)}$ of dimensions $5 \cdot n \times m$, such that the following constraint is satisfied (recall $\mathbf{B}_{L,1}^{(i)}$ represents the first $n$ rows of $\mathbf{B}_L^{(i)}$, $\mathbf{B}_{L,2}^{(i)}$ represents the next $n$ rows of $\mathbf{B}_L^{(i)}$, etc)

$$\sum_{i \,:\, \beta_i = 0} \mathbf{B}_{L,\mathsf{rej}^{(i)}}^{(i)} + \sum_{i \,:\, \beta_i = 1} \mathbf{B}_{L,\mathsf{acc}^{(i)}}^{(i)} = \begin{cases} \mathbf{0} & \text{if } \mathsf{msg} = 0. \\ \sqrt{q} \cdot \left[ \mathbf{I}_n \,||\, \mathbf{0}^{n \times (m-n)} \right] & \text{if } \mathsf{msg} = 1. \end{cases}$$

6. For $i = 1$ to $\ell_{\mathrm{PRG}}$ and $j = 0$ to $L-1$, it chooses $(\mathbf{B}_j^{(i)}, T_j^{(i)}) \leftarrow \mathsf{TrapGen}(1^{5 \cdot n}, 1^m, q)$.
7. Next, it generates the components for each level. For each level $\mathsf{level} \in [1, L]$, do the following:
   (a) Choose matrices $\mathbf{S}_{\mathsf{level}}^{(0)}, \mathbf{S}_{\mathsf{level}}^{(1)} \leftarrow \chi^{n \times n}$ and $\mathbf{E}_{\mathsf{level}}^{(i,0)}, \mathbf{E}_{\mathsf{level}}^{(i,1)} \leftarrow \chi^{5n \times m}$ for $i \leq \ell_{\mathrm{PRG}}$. If either $\mathbf{S}_{\mathsf{level}}^{(0)}$ or $\mathbf{S}_{\mathsf{level}}^{(1)}$ has determinant zero, then set it to be $\mathbf{I}_n$.
   (b) For $b \in \{0,1\}$, set matrix $\mathbf{D}_{\mathsf{level}}^{(i,b)}$ as a permutation of the matrix blocks of $\mathbf{B}_{\mathsf{level}}^{(i)}$ according to the permutation $\sigma_{\mathsf{level},b}^{(i)}(\cdot)$.
   (c) Set $\mathbf{M}_{\mathsf{level}}^{(i,b)} = \left( \mathbf{I}_5 \otimes \mathbf{S}_{\mathsf{level}}^{(b)} \right) \cdot \mathbf{D}_{\mathsf{level}}^{(i,b)} + \mathbf{E}_{\mathsf{level}}^{(i,b)}$ for $i \leq \ell_{\mathrm{PRG}}$.
   (d) Compute $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{\mathsf{level}-1}^{(i)}, T_{\mathsf{level}-1}^{(i)}, \sigma, \mathbf{M}_{\mathsf{level}}^{(i,b)})$
8. The final obfuscated program consists of the LHE evaluation key $\mathsf{ek}$, LHE encryption $\mathsf{ct}$, together with the components $\left( \left\{ \mathbf{B}_{0,1}^{(i)} \right\}_i, \left\{ (\mathbf{C}_j^{(i,0)}, \mathbf{C}_j^{(i,1)}) \right\}_{i,j} \right)$.

Game 1**:** In this experiment, the challenger chooses the matrices $\mathbf{S}_{\mathsf{level}}^{(0)}, \mathbf{S}_{\mathsf{level}}^{(1)}$ without checking if their determinant is non-zero.

7. Next, it generates the components for each level. For each level $\mathsf{level} \in [1, L]$, do the following:

   (a) Choose matrices $\mathbf{S}^{(0)}_{\mathsf{level}}, \mathbf{S}^{(1)}_{\mathsf{level}} \leftarrow \chi^{n \times n}$ and $\mathbf{E}^{(i,0)}_{\mathsf{level}}, \mathbf{E}^{(i,1)}_{\mathsf{level}} \leftarrow \chi^{5n \times m}$ for $i \leq \ell_{\mathrm{PRG}}$.
   (b) For $b \in \{0,1\}$, set matrix $\mathbf{D}^{(i,b)}_{\mathsf{level}}$ as a permutation of the matrix blocks of $\mathbf{B}^{(i)}_{\mathsf{level}}$ according to the permutation $\sigma^{(i)}_{\mathsf{level},b}(\cdot)$.
   (c) Set $\mathbf{M}^{(i,b)}_{\mathsf{level}} = \left(\mathbf{I}_5 \otimes \mathbf{S}^{(b)}_{\mathsf{level}}\right) \cdot \mathbf{D}^{(i,b)}_{\mathsf{level}} + \mathbf{E}^{(i,b)}_{\mathsf{level}}$ for $i \leq \ell_{\mathrm{PRG}}$.
   (d) Compute $\mathbf{C}^{(i,b)}_{\mathsf{level}} \leftarrow \mathsf{SamplePre}(\mathbf{B}^{(i)}_{\mathsf{level}-1}, T^{(i)}_{\mathsf{level}-1}, \sigma, \mathbf{M}^{(i,b)}_{\mathsf{level}})$

**Game 2:** In this experiment, the challenger replaces the pseudorandom string $\beta = \mathrm{PRG}(\alpha)$ with a truly random string.

3. Next, it chooses a uniformly random string $\alpha \leftarrow \{0,1\}^{\ell_{\mathrm{out}}}$ and $\beta \leftarrow \{0,1\}^{\ell_{\mathrm{PRG}}}$.

**Game 3:** In this experiment, the challenger chooses the top level matrices uniformly at random.

5. It chooses $\ell_{\mathrm{PRG}}$ uniformly random matrices $\mathbf{B}^{(i)}_L$ of dimensions $5 \cdot n \times m$, without any constraints.

We will now define $4L+1$ intermediate game $\mathsf{Game}\ (4, j^*, 0), \mathsf{Game}\ (4, j^*, 1), \mathsf{Game}\ (4, j^*, 2), \mathsf{Game}\ (4, j^*, 3)$ for $j^* \in [0, L-1]$ and $\mathsf{Game}\ (4, L, 0)$. The $\mathsf{Game}\ (4, 0, 0)$ will correspond to $\mathsf{Game}\ 3$.

**$\mathsf{Game}\ (4, j^*, 0)$:** In this experiment, the challenger chooses the top $j^*$ level matrices $\mathbf{B}^{(i)}_j$ uniformly at random (without a trapdoor). Also, the top level $\mathbf{C}^{(i,b)}_j$ matrices, for $j > L - j^*$, are chosen randomly from the noise distribution $\chi$.

6. For $i = 1$ to $\ell_{\mathrm{PRG}}$ and $j = 0$ to $L-1$,

   - if $j < L - j^*$ it chooses $(\mathbf{B}^{(i)}_j, T^{(i)}_j) \leftarrow \mathsf{TrapGen}(1^{5 \cdot n}, 1^m, q)$.
   - if $j \geq L - j^*$, it chooses $\mathbf{B}^{(i)}_j$ uniformly at random from $\mathbb{Z}_q^{5 \cdot n \times m}$.

7. Next, it generates the components for each level. For each level $\mathsf{level} \in [1, L - j^*]$, do the following:

   (a) Choose matrices $\mathbf{S}^{(0)}_{\mathsf{level}}, \mathbf{S}^{(1)}_{\mathsf{level}} \leftarrow \chi^{n \times n}$ and $\mathbf{E}^{(i,0)}_{\mathsf{level}}, \mathbf{E}^{(i,1)}_{\mathsf{level}} \leftarrow \chi^{5n \times m}$ for $i \leq \ell_{\mathrm{PRG}}$.
   (b) For $b \in \{0,1\}$, set matrix $\mathbf{D}^{(i,b)}_{\mathsf{level}}$ as a permutation of the matrix blocks of $\mathbf{B}^{(i)}_{\mathsf{level}}$ according to the permutation $\sigma^{(i)}_{\mathsf{level},b}(\cdot)$.
   (c) Set $\mathbf{M}^{(i,b)}_{\mathsf{level}} = \left(\mathbf{I}_5 \otimes \mathbf{S}^{(b)}_{\mathsf{level}}\right) \cdot \mathbf{D}^{(i,b)}_{\mathsf{level}} + \mathbf{E}^{(i,b)}_{\mathsf{level}}$ for $i \leq \ell_{\mathrm{PRG}}$.
   (d) Compute $\mathbf{C}^{(i,b)}_{\mathsf{level}} \leftarrow \mathsf{SamplePre}(\mathbf{B}^{(i)}_{\mathsf{level}-1}, T^{(i)}_{\mathsf{level}-1}, \sigma, \mathbf{M}^{(i,b)}_{\mathsf{level}})$

   For all $\mathsf{level} > L - j^*$, $i \leq \ell_{\mathrm{PRG}}$ and $b \in \{0,1\}$ choose $\mathbf{C}^{(i,b)}_{\mathsf{level}} \leftarrow \chi^{m \times m}$.

**$\mathsf{Game}\ (4, j^*, 1)$:** In this experiment, the challenger chooses $\mathbf{M}^{(i,0)}_{\mathsf{level}}$ uniformly at random for $\mathsf{level} = L - j^*$ ($\mathbf{M}^{(i,1)}_{\mathsf{level}}$ is same as before).

7. Next, it generates the components for each level. For each level $\mathsf{level} \in [1, L - j^*]$, do the following:

   (a) Choose matrices $\mathbf{S}^{(0)}_{\mathsf{level}}, \mathbf{S}^{(1)}_{\mathsf{level}} \leftarrow \chi^{n \times n}$ and $\mathbf{E}^{(i,0)}_{\mathsf{level}}, \mathbf{E}^{(i,1)}_{\mathsf{level}} \leftarrow \chi^{5n \times m}$ for $i \leq \ell_{\mathrm{PRG}}$.
   (b) For $b \in \{0,1\}$, set matrix $\mathbf{D}^{(i,b)}_{\mathsf{level}}$ as a permutation of the matrix blocks of $\mathbf{B}^{(i)}_{\mathsf{level}}$ according to the permutation $\sigma^{(i)}_{\mathsf{level},b}(\cdot)$.

(c) If $\mathsf{level} = L - j^*$, choose $\mathbf{M}_{\mathsf{level}}^{(i,0)} \leftarrow \mathbb{Z}_q^{5\cdot n\times m}$ and $\mathbf{M}_{\mathsf{level}}^{(i,1)}$ same as previous hybrid.
Else set $\mathbf{M}_{\mathsf{level}}^{(i,b)} = \left(\mathbf{I}_5 \otimes \mathbf{S}_{\mathsf{level}}^{(b)}\right) \cdot \mathbf{D}_{\mathsf{level}}^{(i,b)} + \mathbf{E}_{\mathsf{level}}^{(i,b)}$ for $i \le \ell_{\mathrm{PRG}}$.

(d) Compute $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{\mathsf{level}-1}^{(i)}, T_{\mathsf{level}-1}^{(i)}, \sigma, \mathbf{M}_{\mathsf{level}}^{(i,b)})$

For all $\mathsf{level} > L - j^*$, $i \le \ell_{\mathrm{PRG}}$ and $b \in \{0,1\}$ choose $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \chi^{m\times m}$.

**Game $(4, j^*, 2)$:** In this experiment, the challenger chooses $\mathbf{M}_{\mathsf{level}}^{(i,b)}$ uniformly at random for $\mathsf{level} = L - j^*$ and $b \in \{0,1\}$.

7. Next, it generates the components for each level. For each level $\mathsf{level} \in [1, L - j^*]$, do the following:

(a) Choose matrices $\mathbf{S}_{\mathsf{level}}^{(0)}, \mathbf{S}_{\mathsf{level}}^{(1)} \leftarrow \chi^{n\times n}$ and $\mathbf{E}_{\mathsf{level}}^{(i,0)}, \mathbf{E}_{\mathsf{level}}^{(i,1)} \leftarrow \chi^{5n\times m}$ for $i \le \ell_{\mathrm{PRG}}$.

(b) For $b \in \{0,1\}$, set matrix $\mathbf{D}_{\mathsf{level}}^{(i,b)}$ as a permutation of the matrix blocks of $\mathbf{B}_{\mathsf{level}}^{(i)}$ according to the permutation $\sigma_{\mathsf{level},b}^{(i)}(\cdot)$.

(c) If $\mathsf{level} = L - j^*$, choose $\mathbf{M}_{\mathsf{level}}^{(i,b)} \leftarrow \mathbb{Z}_q^{5\cdot n\times m}$.
Else set $\mathbf{M}_{\mathsf{level}}^{(i,b)} = \left(\mathbf{I}_5 \otimes \mathbf{S}_{\mathsf{level}}^{(b)}\right) \cdot \mathbf{D}_{\mathsf{level}}^{(i,b)} + \mathbf{E}_{\mathsf{level}}^{(i,b)}$ for $i \le \ell_{\mathrm{PRG}}$.

(d) Compute $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{\mathsf{level}-1}^{(i)}, T_{\mathsf{level}-1}^{(i)}, \sigma, \mathbf{M}_{\mathsf{level}}^{(i,b)})$

For all $\mathsf{level} > L - j^*$, $i \le \ell_{\mathrm{PRG}}$ and $b \in \{0,1\}$ choose $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \chi^{m\times m}$.

**Game $(4, j^*, 3)$:** In this experiment, the challenger chooses $\mathbf{C}_{L-j^*}^{(i,b)}$ from the noise distribution.

7. Next, it generates the components for each level. For each level $\mathsf{level} \in [1, L - j^*]$, do the following:

(a) Choose matrices $\mathbf{S}_{\mathsf{level}}^{(0)}, \mathbf{S}_{\mathsf{level}}^{(1)} \leftarrow \chi^{n\times n}$ and $\mathbf{E}_{\mathsf{level}}^{(i,0)}, \mathbf{E}_{\mathsf{level}}^{(i,1)} \leftarrow \chi^{5n\times m}$ for $i \le \ell_{\mathrm{PRG}}$.

(b) For $b \in \{0,1\}$, set matrix $\mathbf{D}_{\mathsf{level}}^{(i,b)}$ as a permutation of the matrix blocks of $\mathbf{B}_{\mathsf{level}}^{(i)}$ according to the permutation $\sigma_{\mathsf{level},b}^{(i)}(\cdot)$.

(c) Set $\mathbf{M}_{\mathsf{level}}^{(i,b)} = \left(\mathbf{I}_5 \otimes \mathbf{S}_{\mathsf{level}}^{(b)}\right) \cdot \mathbf{D}_{\mathsf{level}}^{(i,b)} + \mathbf{E}_{\mathsf{level}}^{(i,b)}$ for $i \le \ell_{\mathrm{PRG}}$.

(d) If $\mathsf{level} = L - j^*$, then choose $\mathbf{C}_{L-j^*}^{(i,b)} \leftarrow \chi^{m\times m}$ for $b \in \{0,1\}$.
Else compute $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{\mathsf{level}-1}^{(i)}, T_{\mathsf{level}-1}^{(i)}, \sigma, \mathbf{M}_{\mathsf{level}}^{(i,b)})$

For all $\mathsf{level} > L - j^*$, $i \le \ell_{\mathrm{PRG}}$ and $b \in \{0,1\}$ choose $\mathbf{C}_{\mathsf{level}}^{(i,b)} \leftarrow \chi^{m\times m}$.

**Game 5:** In this experiment, the challenger computes an encryption of $\mathbf{0}^{|P|}$ instead of an encryption of $P$. Note that the LHE secret key $\mathsf{sk}$ is not required at any other step in this experiment. This corresponds to the ideal world.

2. It chooses $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{LHE.Setup}(1^\lambda, 1^{d \log d})$ and sets $\mathsf{ct} \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}, P)$.

# E Predicate Encryption: Key-Policy ABE with Unbounded Decryption Depth

In this section, we construct a predicate encryption (PE) scheme which achieves 1-sided security from any key-policy attribute based encryption (KP-ABE) scheme, a fully homomorphic encryption (FHE) and a lockable obfuscator. Our construction inherits the attribute space and predicate class of the underlying

KP-ABE scheme.

**Outline.** The main idea is to double encrypt the message $m$, i.e. first encrypt the message using the KP-ABE scheme (under attribute $x$) to get ciphertext $\mathsf{ct}_x$, and then encrypt $\mathsf{ct}_x$ using the FHE scheme to compute the final ciphertext $\mathsf{ct}$. At a high level, since the ciphertext $\mathsf{ct}_x$ is encrypted as well, therefore the attribute $x$ is hidden from the adversary. However, to decrypt such a ciphertext, one would require the FHE secret key. Now if the FHE secret key is released in the clear then we could not hope to use FHE security to claim attribute hiding.

To solve this problem, instead of giving the FHE secret key in the clear, we obfuscate the decryption circuit of the FHE scheme with the secret key hardwired in it. Such an obfuscated circuit would take as input an FHE ciphertext, decrypt it inside (using the hardwired secret key) and check against the lock value. Note that we have not yet specified the lock and message used during obfuscation. If we simply use the message $m$ as the lock value, then we do not know how to prove security since the obfuscator guarantees security only if the lock is chosen at random. Therefore, instead of double encrypting the message $m$, we double encrypt a randomly chosen lock value $\alpha$ (first under attribute $x$, then using FHE). And, the message $m$ is set as the output of obfuscated program, i.e. obfuscator is run with lock $\alpha$ and message $m$. The final ciphertext is set as $\mathsf{ct}$ (i.e, the double encryption of lock $\alpha$) and $\widetilde{P}$ (i.e., the obfuscation of decryption circuit). The key generation proceeds identically to that of the underlying ABE scheme. Now decryption proceeds in two phases. First, the inner ciphertext $\mathsf{ct}_x$ (inside ciphertext $\mathsf{ct}$) is homomorphically decrypted using an ABE secret key $\mathsf{sk}_C$ for some predicate $C$. Second, the homomorphically evaluated ciphertext (say $\mathsf{ct}'$) is fed to the obfuscated program $\widetilde{P}$ and output of the program evaluation is simply the output of decryption procedure.

Observe that if $\mathsf{sk}_C$ is a valid secret key (such that $C(x) = 1$), then $\mathsf{ct}'$ would be an FHE encryption of lock $\alpha$ (by correctness of KP-ABE decryption), and output of the obfuscated program with $\mathsf{ct}'$ as input would be message $m$ by correctness of the FHE decryption and program evaluation. At a high level, this provides weak attribute hiding because if the adversary can not decrypt the ciphertext containing the lock (i.e, it does not know the secret key for the corresponding attribute), then it has not no information about the lock value. Therefore, by the obfuscation security, we can guarantee that the challenge ciphertext could be simulated instead. Below we describe our construction $\mathcal{PE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ for attribute spaces $\{\mathcal{X}_\lambda\}_\lambda$, predicate classes $\{\mathcal{C}_\lambda\}_\lambda$ and 1-bit messages.

Let $\mathcal{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.Enc}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Dec})$ be a key-policy attribute based encryption scheme for set of attribute spaces $\{\mathcal{X}_\lambda\}_\lambda$, predicate classes $\{\mathcal{C}_\lambda\}_\lambda$ and message spaces $\{\{0,1\}^{k(\lambda)}\}_\lambda$, and $\mathsf{FHE} = (\mathsf{FHE.Setup}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ be a fully homomorphic encryption scheme for 1-bit messages with decryption circuit of depth $d(\lambda)$ and ciphertexts of length $n(\lambda)$, and $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ be a lockable obfuscator for circuit class $\mathcal{C}_{n\cdot k,k,d}$ (i.e., the class of depth $d(\lambda)$ circuits with $n(\lambda) \cdot k(\lambda)$ bit input and $k(\lambda)$ bit output). The construction follows. For notational convenience, let $k = k(\lambda)$, $n = n(\lambda)$ and $d = d(\lambda)$.

- $\mathsf{Setup}(1^\lambda, \mathcal{X}_\lambda, \mathcal{C}_\lambda) \to (\mathsf{pp}, \mathsf{msk})$. The setup algorithm runs $\mathsf{ABE.Setup}$ to generate public parameters and master secret key as $(\mathsf{abe.pp}, \mathsf{abe.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda, \mathcal{X}_\lambda, \mathcal{C}_\lambda, 1^k)$. It sets $\mathsf{pp} = \mathsf{abe.pp}$ and $\mathsf{msk} = \mathsf{abe.msk}$.

- $\mathsf{Enc}(\mathsf{pp}, x, m) \to \mathsf{ct}$. The encryption algorithm chooses a random message $\alpha \leftarrow \{0,1\}^k$ and computes $\mathsf{ct}_\alpha \leftarrow \mathsf{ABE.Enc}(\mathsf{pp}, x, \alpha)$. Next, it generates an FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$ and encrypts $\mathsf{ct}_\alpha$ as $\mathsf{ct}_{\mathsf{outer}} \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, \mathsf{ct}_\alpha)$.[22] It also obfuscates the FHE decryption circuit (with $\mathsf{fhe.sk}$ hardwired) with message $m$ and lock $\alpha$ as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{FHE.Dec}(\mathsf{fhe.sk}, \cdot), m, \alpha)$.

  Finally, it outputs the ciphertext as $\mathsf{ct} = (\mathsf{fhe.pk}, \mathsf{fhe.ek}, \mathsf{ct}_{\mathsf{outer}}, \widetilde{P})$.

[22] Note that FHE scheme supports bit encryption. Therefore, to encrypt $\mathsf{ct}_\alpha$, which is a multi-bit message, the $\mathsf{FHE.Enc}$ algorithm will be run independently on each bit of $\mathsf{ct}_\alpha$. However, for notational convenience throughout this section we overload the notation and use $\mathsf{FHE.Enc}$ and $\mathsf{FHE.Dec}$ algorithms to encrypt and decrypt multi-bit messages respectively.

- $\mathsf{KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$. The key generation algorithm runs $\mathsf{ABE.KeyGen}$ to generate the secret key as $\mathsf{abe.sk}_C \leftarrow \mathsf{ABE.KeyGen}(\mathsf{msk}, C)$, and outputs $\mathsf{sk}_C = \mathsf{abe.sk}_C$.

- $\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}) \to m$ or $\perp$ . Let $\mathsf{ct} = (\mathsf{fhe.pk}, \mathsf{fhe.ek}, \mathsf{ct}_{\mathsf{outer}}, \widetilde{P})$. The decryption algorithm evaluates the ABE decryption on $\mathsf{ct}_{\mathsf{outer}}$ as $\mathsf{ct}_{\mathsf{inner}} = \mathsf{FHE.Eval}(\mathsf{fhe.ek}, \mathsf{ABE.Dec}(\mathsf{sk}_C, \cdot), \mathsf{ct}_{\mathsf{outer}})$. Next, it evaluates the obfuscated program on input $\mathsf{ct}_{\mathsf{inner}}$, and outputs $\mathsf{Eval}(\widetilde{P}, \mathsf{ct}_{\mathsf{inner}})$.

**Correctness.** For all $\lambda \in \mathbb{N}$, message $m \in \{0,1\}$, attribute $x \in \mathcal{X}_\lambda$, public parameters and master secret key $(\mathsf{abe.pp}, \mathsf{abe.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda, \mathcal{X}_\lambda, \mathcal{C}_\lambda, 1^k)$, the ciphertext corresponding to message $m$ under attribute $x$ in our scheme is of the form $\mathsf{ct} = (\mathsf{fhe.pk}, \mathsf{fhe.ek}, \mathsf{ct}_{\mathsf{outer}}, \widetilde{P})$, where $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$, $\alpha \leftarrow \{0,1\}^k$, $\mathsf{ct}_\alpha \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pp}, x, \alpha)$, $\mathsf{ct}_{\mathsf{outer}} \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, \mathsf{ct}_\alpha)$ and $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{FHE.Dec}(\mathsf{fhe.sk}, \cdot), m, \alpha)$.

For any predicate $C \in \mathcal{C}_\lambda$, the corresponding secret key in our scheme is simply $\mathsf{abe.sk}_C \leftarrow \mathsf{ABE.KeyGen}(\mathsf{abe.msk}, C)$. Let $\mathsf{ct}_{\mathsf{inner}} = \mathsf{FHE.Eval}(\mathsf{fhe.ek}, \mathsf{ABE.Dec}(\mathsf{abe.sk}_C, \cdot), \mathsf{ct}_{\mathsf{outer}})$. Consider the following two cases:

1. $C(x) = 1$ : We know that if $C(x) = 1$, then with all but negligible probability $\mathsf{ABE.Dec}(\mathsf{abe.sk}_C, \mathsf{ct}_\alpha) = \alpha$. Therefore, $\mathsf{ct}_{\mathsf{inner}}$ is an FHE encryption of lock $\alpha$. Or in other words, w.h.p. $\mathsf{FHE.Dec}(\mathsf{fhe.sk}, \mathsf{ct}_{\mathsf{inner}}) = \alpha$. This follows from correctness of the ABE and FHE schemes.

   Next, by correctness of obfuscation, we can conclude that $\mathsf{Eval}(\widetilde{P}, \mathsf{ct}_{\mathsf{inner}}) = m$. Therefore, if $C(x) = 1$, then $\Pr[\mathsf{Dec}(\mathsf{sk}_C = \mathsf{abe.sk}_C, \mathsf{ct}) = m] \geq 1 - \mathrm{negl}(\lambda)$.

2. $C(x) = 0$ : Similarly, we know that if $C(x) = 0$, then with all but negligible probability $\mathsf{ABE.Dec}(\mathsf{abe.sk}_C, \mathsf{ct}_\alpha) = \perp$. Therefore, w.h.p. $\mathsf{FHE.Dec}(\mathsf{fhe.sk}, \mathsf{ct}_{\mathsf{inner}}) = \perp$. This follows from correctness of the ABE and FHE schemes.

   Also by correctness of obfuscation, we can conclude that if $\mathsf{FHE.Dec}(\mathsf{fhe.sk}, \mathsf{ct}_{\mathsf{inner}}) \neq \alpha$, then $\Pr[\mathsf{Eval}(\widetilde{P}, \mathsf{ct}_{\mathsf{inner}}) = \perp] \geq 1 - \mathrm{negl}(\lambda)$. Since $\perp \neq \alpha$, therefore combining these two facts we know that if $C(x) = 0$, then $\Pr[\mathsf{Dec}(\mathsf{sk}_C = \mathsf{abe.sk}_C, \mathsf{ct}) = \perp] \geq 1 - \mathrm{negl}'(\lambda)$ for some negligible function $\mathrm{negl}'(\cdot)$.

Therefore, $\mathcal{PE}$ satisfies the predicate encryption correctness condition.

**Security.** We will now show that the scheme described above achieves 1-sided security as per Definition A.2. Formally, we prove the following.

**Theorem E.1.** If $\mathcal{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.Enc}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Dec})$ is a fully secure attribute based encryption for set of attribute spaces $\{\mathcal{X}_\lambda\}_\lambda$, predicate classes $\{\mathcal{C}_\lambda\}_\lambda$ and message spaces $\{\{0,1\}^{k(\lambda)}\}_\lambda$ satisfying Definition A.1, and $\mathsf{FHE} = (\mathsf{FHE.Setup}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ is a fully homomorphic encryption scheme satisfying Definition 2.4, and $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ is a lockable obfuscator for 1-bit messages and circuit class $\{\mathcal{C}_{n,k,\mathsf{poly}}\}_\lambda$ satisfying Definition 3.4, then $\mathcal{PE}$ is a secure predicate encryption scheme satisfying 1-sided security as per Definition A.3 for 1-bit messages and same attribute space and predicate class as the ABE scheme.

Our proof proceeds via a sequence of hybrid games. Each game is played between the challenger and attacker $\mathcal{A}$. Let $\mathcal{A}$ be any PPT adversary that wins the 1-sided security game with non-negligible advantage. We argue that such an adversary must break security of at least one underlying primitive. The first game corresponds to the 1-sided security game as described in Definition A.2. In the final game, the challenge ciphertext is simulated and does not contain any information about the challenge messages.

We will first define the sequence of hybrid games, and then show that they are computationally indistinguishable.

**Game 1:** In this game, the challenge ciphertext $\mathsf{ct}$ is honestly generated, i.e. first a uniformly random lock $\alpha$ is chosen along with FHE keys $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek})$. Next, it is encrypted under attribute $x$ to get ciphertext $\mathsf{ct}_{\mathsf{inner}}$. Additionally, it encrypts $\mathsf{ct}_{\mathsf{inner}}$ under $\mathsf{fhe.pk}$. Finally, the challenger obfuscates the FHE decryption circuit with secret key $\mathsf{fhe.sk}$ hardwired for lock $\alpha$ and challenge message $m$.

1. **Setup Phase.** The challenger sets up by generating public parameters and master secret key as $(\mathsf{abe.pp}, \mathsf{abe.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda, \mathcal{X}_\lambda, \mathcal{C}_\lambda, 1^k)$. It sends $\mathsf{abe.pp}$ to $\mathcal{A}$.

2. **Pre-Challenge Query Phase.** $\mathcal{A}$ queries the challenger on polynomially many predicate circuits $C_i$ to receive the corresponding secret keys as $\mathsf{sk}_{C_i} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{abe.msk}, C_i)$.

3. **Challenge.** Next, $\mathcal{A}$ sends the challenge messages and attributes $((m_0, x_0), (m_1, x_1))$ to the challenger such that $C_i(x_0) = C_i(x_1) = 0$ for all queried predicates. The challenger chooses a random bit $b \leftarrow \{0,1\}$, a uniformly random lock string $\alpha \leftarrow \{0,1\}^k$, and an FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$. It computes ciphertexts $\mathsf{ct}_{\mathsf{inner}}, \mathsf{ct}_{\mathsf{outer}}$ as $\mathsf{ct}_{\mathsf{inner}} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pp}, x_b, \alpha)$ and $\mathsf{ct}_{\mathsf{outer}} \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, \mathsf{ct}_{\mathsf{inner}})$. It also computes obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{FHE.Dec}(\mathsf{fhe.sk}, \cdot), m_b, \alpha)$, and sends $(\mathsf{fhe.pk}, \mathsf{fhe.ek}, \mathsf{ct}_{\mathsf{outer}}, \widetilde{P})$ to $\mathcal{A}$.

4. **Post-Challenge Query Phase.** $\mathcal{A}$ queries the challenger on polynomially many predicate circuits $C_i$ as before. The challenger handles these as in pre-challenge query phase.

5. **Guess.** $\mathcal{A}$ outputs it guess $b'$ and wins if $b = b'$.

**Game 2:** This is same as Game 1, except the challenger computes ciphertext $\mathsf{ct}_{\mathsf{inner}}$ as an encryption of an all zero string instead of $\alpha$.

3. **Challenge.** Next, $\mathcal{A}$ sends the challenge messages and attributes $((m_0, x_0), (m_1, x_1))$ to the challenger such that $C_i(x_0) = C_i(x_1) = 0$ for all queried predicates. The challenger chooses a random bit $b \leftarrow \{0,1\}$, a uniformly random lock string $\alpha \leftarrow \{0,1\}^k$, and an FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$. It computes ciphertexts $\mathsf{ct}_{\mathsf{inner}}, \mathsf{ct}_{\mathsf{outer}}$ as $\mathsf{ct}_{\mathsf{inner}} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pp}, x_b, 0^k)$ and $\mathsf{ct}_{\mathsf{outer}} \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, \mathsf{ct}_{\mathsf{inner}})$. It also computes obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{FHE.Dec}(\mathsf{fhe.sk}, \cdot), m_b, \alpha)$, and sends $(\mathsf{fhe.pk}, \mathsf{fhe.ek}, \mathsf{ct}_{\mathsf{outer}}, \widetilde{P})$ to $\mathcal{A}$.

**Game 3:** This is same as Game 2, except the challenger does not choose the lock $\alpha$ anymore and it simulates the obfuscated program $\widetilde{P}$ instead of generating it honestly as an obfuscation of the FHE decryption circuit.

3. **Challenge.** Next, $\mathcal{A}$ sends the challenge messages and attributes $((m_0, x_0), (m_1, x_1))$ to the challenger such that $C_i(x_0) = C_i(x_1) = 0$ for all queried predicates. The challenger chooses a random bit $b \leftarrow \{0,1\}$ and an FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$. It computes ciphertexts $\mathsf{ct}_{\mathsf{inner}}, \mathsf{ct}_{\mathsf{outer}}$ as $\mathsf{ct}_{\mathsf{inner}} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pp}, x_b, 0^k)$ and $\mathsf{ct}_{\mathsf{outer}} \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, \mathsf{ct}_{\mathsf{inner}})$. It also computes obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathcal{O}.\mathsf{Sim}(1^\lambda, 1^s, 1)$ (where $s = |\mathsf{FHE.Dec}(\mathsf{fhe.sk}, \cdot)|$), and sends $(\mathsf{fhe.pk}, \mathsf{fhe.ek}, \mathsf{ct}_{\mathsf{outer}}, \widetilde{P})$ to $\mathcal{A}$.

**Game 4:** This is same as Game 3, except the challenger computes ciphertext $\mathsf{ct}_{\mathsf{outer}}$ as an encryption of an all zero string instead of $\mathsf{ct}_{\mathsf{inner}}$.

3. **Challenge.** Next, $\mathcal{A}$ sends the challenge messages and attributes $((m_0, x_0), (m_1, x_1))$ to the challenger such that $C_i(x_0) = C_i(x_1) = 0$ for all queried predicates. The challenger chooses a random bit $b \leftarrow \{0,1\}$ and an FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$. It computes ciphertexts $\mathsf{ct}_{\mathsf{outer}}$ as $\mathsf{ct}_{\mathsf{outer}} \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, 0^\ell)$, where $\ell$ be the length of ABE ciphertexts. It also computes obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathcal{O}.\mathsf{Sim}(1^\lambda, 1^s, 1)$, and sends $(\mathsf{fhe.pk}, \mathsf{fhe.ek}, \mathsf{ct}_{\mathsf{outer}}, \widetilde{P})$ to $\mathcal{A}$.

Let $\mathsf{Adv}^i_{\mathcal{A}} = |\Pr[b' = b] - 1/2|$ denote the advantage of adversary $\mathcal{A}$ in guessing the bit $b$ in Game $i$. First, note that $\mathsf{Adv}^4_{\mathcal{A}} = 0$. In other words, $\mathcal{A}$'s advantage in Game 4 is 0. This is because the challenge ciphertext $(\mathsf{fhe.pk}, \mathsf{fhe.ek}, \mathsf{ct}_{\mathsf{outer}}, \widetilde{P})$ does not contain any information about bit $b$.

To complete the proof, we establish via a sequence of lemmas that no PPT adversary $\mathcal{A}$ can distinguish between each adjacent game with non-negligible probability. We show via a sequence of lemmas that $|\mathsf{Adv}^i_{\mathcal{A}} - \mathsf{Adv}^{i+1}_{\mathcal{A}}|$ is negligible for all $i = 1, 2, 3$. Below we discuss our lemmas in detail.

**Lemma E.1.** If $\mathcal{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.Enc}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Dec})$ is a fully secure attribute based encryption, then for all PPT adversaries $\mathcal{A}$, $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is negligible in the security parameter $\lambda$.

*Proof.* Suppose there exists an adversary $\mathcal{A}$ such that $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is non-negligible. We construct an algorithm $\mathcal{B}$ that can distinguish encryptions of lock $\alpha$ and an all zeros string, therefore break security of the ABE scheme.

The ABE challenger generates a key pair $(\mathsf{abe.pp}, \mathsf{abe.sk})$ and sends $\mathsf{abe.pp}$ to $\mathcal{B}$. $\mathcal{B}$ simply forwards the public parameters $\mathsf{abe.pp}$ to adversary $\mathcal{A}$. $\mathcal{A}$ queries the reduction algorithm $\mathcal{B}$ on polynomially many predicates $C_i$ for corresponding secret keys. $\mathcal{B}$ forwards each predicate query $C_i$ to the ABE challenger and receives back secret key $\mathsf{sk}_{C_i}$, which it then sends to $\mathcal{A}$ as its response. Next, $\mathcal{A}$ sends the challenge message-attribute pairs $((m_0, x_0), (m_1, x_1))$ to $\mathcal{B}$ such that $C_i(x_0) = C_i(x_1) = 0$ for all queried predicates. $\mathcal{B}$ chooses a random string $\alpha \leftarrow \{0,1\}^k$ and bit $b \leftarrow \{0,1\}$, and sends $t_0 = \alpha$ and $t_1 = 0^k$ as its challenge messages and $x_b$ as the challenge attribute to the ABE challenger. The ABE challenger chooses a random bit $\beta \leftarrow \{0,1\}$, computes the challenge ciphertext $\mathsf{ct}^* \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pp}, x_b, t_\beta)$ and sends $\mathsf{ct}^*$ to $\mathcal{B}$. After receiving $\mathsf{ct}^*$ from the challenger, $\mathcal{B}$ runs Step 3 as in Game 1. That is, $\mathcal{B}$ chooses an FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$ and computes ciphertext $\mathsf{ct}_{\mathsf{outer}}$ as $\mathsf{ct}_{\mathsf{outer}} \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, \mathsf{ct}^*)$. It also computes obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{FHE.Dec}(\mathsf{fhe.sk}, \cdot), m_b, \alpha)$, and sends $(\mathsf{fhe.pk}, \mathsf{fhe.ek}, \mathsf{ct}_{\mathsf{outer}}, \widetilde{P})$ to $\mathcal{A}$. Next, $\mathcal{A}$ makes more key queries and $\mathcal{B}$ answers them as before by forwarding those to ABE challenger. Finally, $\mathcal{A}$ outputs its guess $b'$. If $b = b'$, then $\mathcal{B}$ sends 0 as its guess (i.e., $\alpha$ was encrypted), otherwise it sends 1 as its guess (i.e., $0^k$ was encrypted) to the ABE challenger.

Note that if the ABE challenger encrypted $\alpha$ (i.e., $\beta = 0$), then $\mathcal{B}$ perfectly simulates Game 1 for adversary $\mathcal{A}$. Otherwise it simulates Game 2 for $\mathcal{A}$. As a result, if $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is non-negligible, then $\mathcal{B}$ breaks the ABE scheme's security with non-negligible advantage. ∎

**Lemma E.2.** If $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ is a secure lockable obfuscator, then for all PPT adversaries $\mathcal{A}$, $|\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}}|$ is negligible in the security parameter $\lambda$.

*Proof.* Suppose there exists an adversary $\mathcal{A}$ such that $|\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}}|$ is non-negligible. We construct an algorithm $\mathcal{B}$ that can distinguish an obfuscation of the FHE decryption circuit with a random lock from a simulated obfuscated program, therefore break security of the obfuscation scheme.

$\mathcal{B}$ samples an ABE key pair $(\mathsf{abe.pp}, \mathsf{abe.sk})$ and sends $\mathsf{abe.pp}$ to adversary $\mathcal{A}$. $\mathcal{A}$ queries the reduction algorithm $\mathcal{B}$ on polynomially many predicates $C_i$ for corresponding secret keys. $\mathcal{B}$ answers each query with secret key $\mathsf{sk}_{C_i}$, with $\mathsf{sk}_{C_i} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{abe.msk}, C_i)$. Next, $\mathcal{A}$ sends the challenge message-attribute pairs $((m_0, x_0), (m_1, x_1))$ to $\mathcal{B}$ such that $C_i(x_0) = C_i(x_1) = 0$ for all queried predicates. $\mathcal{B}$ chooses a random bit $b \leftarrow \{0,1\}$ and an FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$, and computes ciphertext $\mathsf{ct}_{\mathsf{inner}}, \mathsf{ct}_{\mathsf{outer}}$ as $\mathsf{ct}_{\mathsf{inner}} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pp}, x_b, 0^k)$ and $\mathsf{ct}_{\mathsf{outer}} \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, \mathsf{ct}_{\mathsf{inner}})$. $\mathcal{B}$ sends the $\mathsf{FHE.Dec}(\mathsf{fhe.sk}, \cdot)$ circuit along with message $m_b$ to the obfuscation challenger, and receives an obfuscated program $\widetilde{P}$. $\mathcal{B}$ finally sends $(\mathsf{fhe.pk}, \mathsf{fhe.ek}, \mathsf{ct}_{\mathsf{outer}}, \widetilde{P})$ to $\mathcal{A}$ as the challenge ciphertext. Next, $\mathcal{A}$ makes more key queries and $\mathcal{B}$ answers them as before. Finally, $\mathcal{A}$ outputs its guess $b'$. If $b = b'$, then $\mathcal{B}$ sends 0 as its guess (i.e., FHE decryption circuit was obfuscated), otherwise it sends 1 as its guess (i.e., program $\widetilde{P}$ was simulated) as its guess to the obfuscation challenger.

Note that if the obfuscation challenger obfuscated $\mathsf{FHE.Dec}(\mathsf{fhe.sk}, \cdot)$ for some lock $\alpha$, then $\mathcal{B}$ perfectly simulates Game 2 for adversary $\mathcal{A}$. Otherwise it simulates Game 3 for $\mathcal{A}$. As a result, if $|\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}}|$ is non-negligible, then $\mathcal{B}$ breaks the obfuscation scheme's security with non-negligible advantage. ∎

**Lemma E.3.** If $\mathsf{FHE} = (\mathsf{FHE.Setup}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ is a fully homomorphic encryption scheme, then for all PPT adversaries $\mathcal{A}$, $|\mathsf{Adv}^3_{\mathcal{A}} - \mathsf{Adv}^4_{\mathcal{A}}|$ is negligible in the security parameter $\lambda$.

*Proof.* Suppose there exists an adversary $\mathcal{A}$ such that $|\mathsf{Adv}^3_{\mathcal{A}} - \mathsf{Adv}^4_{\mathcal{A}}|$ is non-negligible. We construct an algorithm $\mathcal{B}$ that can distinguish encryptions of ciphertext $\mathsf{ct}_{\mathsf{inner}}$ and an all zeros string, therefore break security of the FHE scheme.

The FHE challenger generates a key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}, \mathsf{fhe.ek})$ and sends $\mathsf{fhe.pk}, \mathsf{fhe.ek}$ to $\mathcal{B}$. $\mathcal{B}$ samples an ABE key pair $(\mathsf{abe.pp}, \mathsf{abe.sk})$ and sends $\mathsf{abe.pp}$ to adversary $\mathcal{A}$. $\mathcal{A}$ queries the reduction algorithm $\mathcal{B}$ on polynomially many predicates $C_i$ for corresponding secret keys. $\mathcal{B}$ answers each query with secret key $\mathsf{sk}_{C_i}$, with $\mathsf{sk}_{C_i} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{abe.msk}, C_i)$. Next, $\mathcal{A}$ sends the challenge message-attribute pairs $((m_0, x_0), (m_1, x_1))$ to $\mathcal{B}$ such that $C_i(x_0) = C_i(x_1) = 0$ for all queried predicates. $\mathcal{B}$ chooses a random bit $b \leftarrow \{0,1\}$ and computes ciphertext $\mathsf{ct}_{\mathsf{inner}}$ as $\mathsf{ct}_{\mathsf{inner}} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pp}, x_b, 0^k)$, and sends $t_0 = \mathsf{ct}_{\mathsf{inner}}$ and $t_1 = 0^\ell$ as its challenge messages to the FHE challenger. The FHE challenger chooses a random bit $\beta \leftarrow \{0,1\}$, computes the challenge ciphertext $\mathsf{ct}^* \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, t_\beta)$ and sends $\mathsf{ct}^*$ to $\mathcal{B}$. After receiving $\mathsf{ct}^*$ from the challenger, $\mathcal{B}$ computes obfuscated program $\widetilde{P}$ as $\widetilde{P} \leftarrow \mathcal{O}.\mathsf{Sim}(1^\lambda, 1^s, 1)$, and sends $(\mathsf{fhe.pk}, \mathsf{fhe.ek}, \mathsf{ct}^*, \widetilde{P})$ to $\mathcal{A}$. Next, $\mathcal{A}$ makes more key queries and $\mathcal{B}$ answers them as before. Finally, $\mathcal{A}$ outputs its guess $b'$. If $b = b'$, then $\mathcal{B}$ sends 0 as its guess (i.e., $\mathsf{ct}_{\mathsf{inner}}$ was encrypted), otherwise it sends 1 (i.e., $0^\ell$ was encrypted) as its guess to the FHE challenger.

Note that if the FHE challenger encrypted $\mathsf{ct}_{\mathsf{inner}}$ (i.e., $\beta = 0$), then $\mathcal{B}$ perfectly simulates Game 3 for adversary $\mathcal{A}$. Otherwise it simulates Game 4 for $\mathcal{A}$. Therefore, if $|\mathsf{Adv}^3_{\mathcal{A}} - \mathsf{Adv}^4_{\mathcal{A}}|$ is non-negligible, then $\mathcal{B}$ breaks the FHE scheme's security with non-negligible advantage. ■

# F Predicate Encryption: SIM-1-Sided Security

In this section, we transform a PE scheme from one which achieves 1-sided security to another scheme that achieves SIM-1-sided security. The transformation inherits the message space of the underlying PE scheme. However, the attribute space and predicate class are slightly smaller than original.

Let $\mathsf{PE} = (\mathsf{PE.Setup}, \mathsf{PE.Enc}, \mathsf{PE.KeyGen}, \mathsf{PE.Dec})$ be a predicate encryption scheme for set of attribute spaces $\{\mathcal{X}_\lambda\}_\lambda$, predicate classes $\{\mathcal{C}_\lambda\}_\lambda$ and message spaces $\{\mathcal{M}_\lambda\}_\lambda$. Below we describe our construction.

For each boolean predicate $C : \{0,1\}^n \to \{0,1\}$, we use $\widetilde{C}$ to denote the boolean predicate on $n+1$ bits such that $\widetilde{C}(x) = 1$ iff $x_1 = 1 \wedge C(x_2 \,||\, \cdots \,||\, x_{n+1})$, where $x_i$ is $i^{th}$ bit of $x$. In other words, $\widetilde{C}$ evaluates to 1 iff the first input bit is 1 and predicate $C$ outputs 1 on remaining bits as input. Similarly, for a predicate class $\mathcal{C}_\lambda$, we use $\widetilde{\mathcal{C}}_\lambda$ to denote the predicate class that contains predicate $\widetilde{C}$ for each predicate $C \in \mathcal{C}_\lambda$.

- $\mathsf{Setup}(1^\lambda, 1^\ell, \mathcal{C}_\lambda, \mathcal{M}_\lambda) \to (\mathsf{pp}, \mathsf{msk})$. The setup algorithm runs $\mathsf{PE.Setup}$ to generate public parameters and master secret key as $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{PE.Setup}(1^\lambda, 1^{\ell+1}, \widetilde{\mathcal{C}}_\lambda, \mathcal{M}_\lambda)$.
- $\mathsf{Enc}(\mathsf{pp}, x, m) \to \mathsf{ct}$. The encryption algorithm computes ciphertext $\mathsf{ct}$ as $\mathsf{ct} \leftarrow \mathsf{PE.Enc}(\mathsf{pp}, 1 \,||\, x, m)$.
- $\mathsf{KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$. The key generation algorithm runs $\mathsf{PE.KeyGen}$ to generate the secret key as $\mathsf{sk}_C \leftarrow \mathsf{PE.KeyGen}(\mathsf{msk}, \widetilde{C})$.
- $\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}) \to m$. The decryption algorithm runs $\mathsf{PE.Dec}$ and outputs $m = \mathsf{PE.Dec}(\mathsf{sk}_C, \mathsf{ct})$.

**Correctness.** The correctness of above scheme follows directly from the correctness of underlying PE scheme.

**Security.** We will now show that the scheme described above achieves 1-sided security as per Definition A.2. Formally, we prove the following.

**Theorem F.1.** If $\mathsf{PE} = (\mathsf{PE.Setup}, \mathsf{PE.Enc}, \mathsf{PE.KeyGen}, \mathsf{PE.Dec})$ is a secure predicate encryption scheme satisfying 1-sided security as per Definition A.2 for set of attribute spaces $\left\{\{0,1\}^{\ell(\lambda)+1}\right\}_\lambda$, predicate classes $\left\{\widetilde{\mathcal{C}}_\lambda\right\}_\lambda$ and message spaces $\{\mathcal{M}_\lambda\}_\lambda$ satisfying Definition A.2, then $\mathcal{PE}$ is a secure predicate encryption scheme satisfying SIM-1-sided security as per Definition A.3 for set of attribute spaces $\left\{\{0,1\}^{\ell(\lambda)}\right\}_\lambda$, predicate classes $\{\mathcal{C}_\lambda\}_\lambda$ and message spaces $\{\mathcal{M}_\lambda\}_\lambda$.

We start by sketching the simulator $\mathsf{Sim}$. Let $\mathcal{A}$ be any real world adversary. On input the security parameter $\lambda$, public parameters $\mathsf{pp}$, attribute length $\ell + 1$, the simulator runs as follows.

- $\mathsf{Sim}$ encrypts all 0's string as the message under the all 0's string as an attribute. Concretely, it outputs $\mathsf{ct} = \mathsf{PE.Enc}(\mathsf{pp}, 0^{|m|}, 0^{\ell+1})$.

Let $p^1_{\mathcal{A}}$ denote the probability that adversary $\mathcal{A}$ outputs 1 in the 1-sided security game as described in Definition A.2 and $p^2_{\mathcal{A}}$ denote the probability that adversary $\mathcal{A}$ outputs 1 in simulated execution (i.e., when the challenge ciphertext is simulated). We show that $|p^1_{\mathcal{A}} - p^2_{\mathcal{A}}|$ is negligible.

**Lemma F.1.** If $\mathsf{PE} = (\mathsf{PE.Setup}, \mathsf{PE.Enc}, \mathsf{PE.KeyGen}, \mathsf{PE.Dec})$ is a secure predicate encryption scheme satisfying 1-sided security, then for all PPT adversaries $\mathcal{A}$, $|p^1_{\mathcal{A}} - p^2_{\mathcal{A}}|$ is negligible in the security parameter $\lambda$.

*Proof.* Suppose there exists an adversary $\mathcal{A}$ such that $|p^1_{\mathcal{A}} - p^2_{\mathcal{A}}|$ is non-negligible. We construct an algorithm $\mathcal{B}$ that can distinguish encryptions of challenge message $m$ under attribute $x$ and an all zeros string under all zeros attribute string, therefore break 1-sided security of the PE scheme.

The PE challenger generates a key pair $(\mathsf{pp}, \mathsf{msk})$ and sends $\mathsf{pp}$ to $\mathcal{B}$. $\mathcal{B}$ simply forwards the public parameters $\mathsf{pp}$ to adversary $\mathcal{A}$. $\mathcal{A}$ queries the reduction algorithm $\mathcal{B}$ on polynomially many predicates $C_i$ for corresponding secret keys. $\mathcal{B}$ queries predicate $\widetilde{C_i}$ to the PE challenger and receives back secret key $\mathsf{sk}_{C_i}$, which it then sends to $\mathcal{A}$ as its response. Next, $\mathcal{A}$ sends the challenge message-attribute pair $(m, x)$ to $\mathcal{B}$ such that $C_i(x) = 0$ for all queried predicates. $\mathcal{B}$ sends $((m, 1 \,||\, x), (0^{|m|}, 0^{\ell+1}))$ as its challenge message-attribute pairs to the PE challenger. The PE challenger chooses a random bit $b \leftarrow \{0, 1\}$, computes the challenge ciphertext $\mathsf{ct} \leftarrow \mathsf{PE.Enc}(\mathsf{pp}, x_b, m_b)$ (where $x_0 = x, m_0 = m, x_1 = 0^{\ell+1}, m_1 = 0^{|m|}$), and sends $\mathsf{ct}$ to $\mathcal{B}$. $\mathcal{B}$ forwards $\mathsf{ct}$ as the challenge ciphertext to $\mathcal{A}$. Next, $\mathcal{A}$ makes more key queries and $\mathcal{B}$ answers them as before by forwarding those to the PE challenger. Finally, $\mathcal{A}$ outputs a bit $b'$ and $\mathcal{B}$ outputs $b'$ as its guess to the PE challenger.

We would like to point out that all the predicate queries made by $\mathcal{A}$ can be forwarded to the PE challenger because by definition $\widetilde{C}(0^{\ell+1}) = 0$ for all predicates $C$. Also, note that if the PE challenger encrypted $m$ under attribute $x$ (i.e., $b = 0$), then $\mathcal{B}$ perfectly simulates the 1-sided security game for adversary $\mathcal{A}$. Otherwise it simulates the challenge ciphertext for $\mathcal{A}$. As a result, if $|p^1_{\mathcal{A}} - p^2_{\mathcal{A}}|$ is non-negligible, then $\mathcal{B}$ breaks the PE scheme's security with non-negligible advantage. ∎