Impossibility of Simulation Secure Functional Encryption Even with Random Oracles

Shashank Agrawal[1], Venkata Koppula[2], and Brent Waters[2⋆]

[1] Visa Research
shashank.agraval@gmail.com
[2] University of Texas at Austin
{kvenkata, bwaters}@cs.utexas.edu

Abstract. In this work we study the feasibility of achieving simulation security in functional encryption (FE) in the random oracle model. Our main result is negative in that we give a functionality for which it is impossible to achieve simulation security even with the aid of random oracles.

We begin by giving a formal definition of simulation security that explicitly incorporates the random oracles. Next, we show a particular functionality for which it is impossible to achieve simulation security. Here messages are interpreted as seeds to a (weak) pseudorandom function family F and private keys are ascribed to points in the domain of the function. On a message s and private key x one can learn $F(s, x)$. We show that there exists an attacker that makes a polynomial number of private key queries followed by a single ciphertext query for which there exists no simulator.

Our functionality and attacker access pattern closely matches the *standard model* impossibility result of Agrawal, Gorbunov, Vaikuntanathan and Wee (CRYPTO 2013). The crux of their argument is that no simulator can succinctly program in the outputs of an unbounded number of evaluations of a pseudorandom function family into a fixed size ciphertext. However, their argument does not apply in the random oracle setting since the oracle acts as an additional conduit of information which the simulator can program. We overcome this barrier by proposing an attacker who decrypts the challenge ciphertext with the secret keys issued earlier *without* using the random oracle, even though the decryption algorithm may require it. This involves collecting most of the useful random oracle queries in advance, without giving the simulator too many opportunities to program.

On the flip side, we demonstrate the utility of the random oracle in simulation security. Given only public key encryption and low-depth PRGs we show how to build an FE system that is simulation secure for any poly-time attacker that makes an unbounded number of message queries, but an a-priori bounded number of key queries. This bests what is possible in the standard model where it is only feasible to achieve security for an attacker that is bounded both in the number of key and message

⋆ Supported by NSF CNS-1414082, DARPA SafeWare, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

queries it makes. We achieve this by creating a system that leverages the random oracle to get one-key security and then adapt previously known techniques to boost the system to resist up to q queries.

Finally, we ask whether it is possible to achieve simulation security for an unbounded number of messages and keys, but where all key queries are made *after* the message queries. We show this too is impossible to achieve using a different twist on our first impossibility result.

1 Introduction

The traditional notion of public key encryption systems provide "all or nothing" semantics regarding encrypted data. In such a system a message m is encrypted under a public key, pk, to produce a ciphertext ct. A user that holds the corresponding secret key can decrypt ct and learn the entire message m, while any other user will not learn anything about the contents of the message. The work of Sahai and Waters [32] conceived cryptosystems that moved beyond these limited semantics to ones where a private key would give a select view of encrypted data. These efforts [32, 13, 25] cumulated in the concept of functional encryption. In a functional encryption system an authority will generate a pair of a public key and master key pair $(\mathsf{pk}, \mathsf{msk})$. Any user can encrypt a ciphertext ct using the public key, while the authority can use the master secret key msk to generate a secret key sk_f that is tied to the functionality f. A holder of sk_f can use it to decrypt a ciphertext ct, but instead of learning the message m, the decryptor's decryption will instead output $f(m)$.

One challenge in defining and designing functional encryption (FE) systems is in finding a definition to capture security. The earliest formal definitions of functional encryption [13, 25] (back when the terminology of "predicate encryption" was used) defined security in terms of an indistinguishability game. Briefly, a system is indistinguishability secure if no poly-time attacker that receives secret keys for functions $f_1, \ldots, f_Q$ can distinguish between encryptions of m_0, m_1 so long as $f_i(m_0) = f_i(m_1)$ for $i = 1, \ldots, Q$.

Subsequent works [12, 29, 5, 2] aimed to capture various notions of simulation-based security. To achieve simulation one must be able to show that for each attacker there exists a poly-time simulator $\mathcal{S}$ that can produce a transcript that emulates the attacker's real world view, but when only given access to what the evaluation of the secret key functions $f(\cdot)$ were on the attacker's messages. (We will return to describing simulation-based security in more detail shortly.) While these simulation definitions had the appeal of perhaps capturing a stronger notion of security than the indistinguishability-based ones, they were limited in that multiple works [12, 29, 5, 2, 22] showed that this notion is impossible to achieve in the standard model for even very basic functionalities such as identity-based encryption [33, 11]. The only exception being in the restricted case where the attacker is only allowed to access an a-priori bounded number of secret keys [20].

While these results essentially put a hard stop on realizing (collusion-resistant) simulation security in the standard model, the door to leveraging the random or-

acle model [6] still remained wide open. Notably, Boneh, Sahai and Waters [12] building on techniques from non-committing encryption [28] showed that the random oracle could be leveraged to turn any indistinguishability secure public index FE scheme into one that was simulation secure. Recall that a public index scheme is one where an encrypted message is split into a hidden payload and a non-hidden index and the secret key operates only on the index. The set of such schemes includes identity-based encryption [33, 11] and attribute-based encryption [32]. Thus, they showed that introducing a random oracle was enough to circumvent their own standard model IBE result. In this work we wish to understand what are the possibilities and limitations (if any) for using random oracles to achieve simulation security in FE systems. Our work begins with the question:

Is it possible to achieve simulation secure functional encryption for any functionality in the random oracle model?

Our main result is to show that there exist functionalities for which there cannot exist a simulation secure functional encryption system *even* in the random oracle model.

On the flip side, we demonstrate the utility of the random oracle in simulation security. Given only public key encryption and low-depth PRGs we show how to build an FE system that is simulation secure for any poly-time attacker that makes an unbounded number of message queries, but an a-priori bounded number of key queries. This beats what is possible in the standard model where it is only feasible to achieve security for an attacker that is bounded both in the number of key and message queries it makes. We achieve this by creating a system that leverages the random oracle to get one-key security and then adapt previously known techniques to boost the system to resist up to q queries.

Finally, we ask whether it is possible to achieve simulation security for an unbounded number of messages and keys, but where all key queries are made *after* the message queries. We show this too is impossible to achieve by repurposing our main impossibility result to the new setting.

1.1 Our Main Impossibility Result

We show the impossibility result for the case where messages are interpreted as keys or seeds to a (weak) Pseudo Random Function (PRF) [18] family and secret keys are points in the domain of the PRF. Agrawal, Gorbunov, Vaikuntanathan and Wee [2] showed that such a functionality could not be simulation secure in the standard model. Here we show that this limitation holds even with the introduction of random oracles.

We begin our exposition by describing the definition of simulation security in a little more depth and briefly overviewing the AGVW impossibility analysis.

Simulation security. Simulation security for FE is defined by means of real and ideal experiments. In the real experiment, an adversary $\mathcal{A}$ gets secret keys for

functions f and ciphertexts for challenge messages m of its choice. The secret key queries can either be sent before the challenge messages (also referred to as pre-challenge queries) or after the challenge messages (post challenge queries). In the ideal world, on the other hand, a simulator $\mathcal{S}$ needs to generate challenge ciphertexts and keys given only the minimal information. In particular, when $\mathcal{A}$ requests that a challenge message m be encrypted, $\mathcal{S}$ only gets $f(m)$ on all the pre-challenge functions f queried by $\mathcal{A}$ (instead of m itself), and must generate a ciphertext that $\mathcal{A}$ cannot distinguish from the one in the real world. Similarly, when $\mathcal{A}$ makes a post-challenge key query for f', $\mathcal{S}$ must generate a secret key given just f', $f'(m)$ for all challenge messages m.

An FE scheme is $(q_{\mathsf{pre}}, q_{\mathsf{chal}}, q_{\mathsf{post}})$-simulation secure if it can withstand adversaries that make at most q_{pre} pre-challenge key queries, q_{chal} challenge encryption requests, and q_{post} post-challenge key queries. Ideally, one would like to capture all polynomial-time adversaries, who can make any number of queries they want. However, even simple functionalities like identity-based encryption do not have a scheme secure against an arbitrary number of encryption requests followed by one key query, i.e., IBE does not have a $(0, \mathsf{poly}, 1)$-simulation secure scheme [12, 5] in the standard model. Here poly denotes that any number of encryption requests can be made, as long as there is a polynomial bound on them.

AGVW impossibility. A different kind of impossibility was shown by Agrawal et al. [2]. They interpret messages as seeds to a weak pseudorandom family wPRF[3] and secret keys as points in the domain of the family. When a ciphertext for s is decrypted with a secret key for x, the output is $\mathsf{wPRF}(s, x)$. They show that there does not exist a simulation-secure FE scheme for this family that can tolerate adversaries which can make an arbitrary number of pre-challenge key queries and then request for the encryption of just one message (i.e., $(\mathsf{poly}, 1, 0)$-simulation security). Intuitively, when the adversary outputs a message s in the ideal world, the simulator gets $\mathsf{wPRF}(s, x_1), \ldots, \mathsf{wPRF}(s, x_q)$ (if q is the number of post-challenge key queries), which is computationally indistinguishable from q uniformly random strings. The simulator must output a ciphertext ct now that decrypts correctly with all the keys issued before. Note that when the keys were issued, simulator had no information about s, so it must somehow *compress* q random strings into ct. However, as Agrawal et al. show, the output of a pseudorandom function family is *incompressible*. Thus, by choosing a large enough q, they arrive at the impossibility result.

Random oracle model. In the random oracle model though, Agrawal et al.'s impossibility argument breaks down. Informally speaking, the random oracle acts as an additional conduit of information which the simulator can program even *after* ct appears. For instance, if the decryption algorithm makes RO queries, then the simulator could program such queries when adversary tries to decrypt ct

[3] A weak pseudorandom function family provides security only against attackers that do not get to choose the points at the which the PRF is evaluated. These points are chosen randomly by the challenger.

with the secret keys issued earlier. Indeed, Boneh et al. show that their $(0, \mathsf{poly}, 1)$ impossibility for IBE can be circumvented by employing RO in the encryption and decryption algorithms.

Thus we need a very different approach. We would like to build an adversary $\mathcal{A}^\star$ that "cuts off" RO in the decryption process, and is able to work without it. This involves a delicate balancing act between cutting off too early and too late. In one extreme case, if $\mathcal{A}^\star$ does not invoke RO at all and makes up its own responses, then these would not match with the actual RO responses in encryption and key generation. Thus decryption would always fail in both the real and ideal worlds, and there will be no distinction between them. On the other extreme, if $\mathcal{A}^\star$ just used the RO all the way through, it would provide the simulator enough opportunity to program in the desired information. (As a result, we will not be able to use the incompressibility of wPRF.)

At a high level, our approach is similar to the Impagliazzo-Rudich "heavy-query" algorithm [23]. First, there is an initial learning phase where $\mathcal{A}^\star$ will build a list of "high frequency" random oracle queries and responses associated with each secret key and the challenge ciphertext. Later the attacker will be able to use this list to replace the use of the actual random oracle during decryption. If some query is not found in the list, then $\mathcal{A}^\star$ will choose a random value for it on its own. Informally, we get the following result:

Theorem 1 (Main Theorem, informal). *There does not exist a* $(\mathsf{poly}, 1, 0)$*-simulation secure FE scheme for the class of (weak) pseudo-random functions in the random oracle model.*

Related work. This bears a resemblance to the work of Canetti, Kalai and Paneth [15] who show impossibility of VBB obfuscation even with ROs. In their case they show that any obfuscated program that uses the RO can be translated into one that does not need it. They do this by collecting the frequently used RO queries and bundling this with the core obfuscated code. On one hand, these queries do not give any information about the program, but on the other, result in an obfuscation that is only approximately correct. Such imperfect correctness, however, is enough to invoke the impossibility of Bitansky and Paneth [9].

One might ask if we can show whether RO can be dispensed with in any simulation secure FE in a similar way. If we could establish this, then prior impossibility results [12, 5, 2] would imply RO impossibility as well. The answer to this is negative as we recall that Boneh, Sahai and Waters [12] showed specific functionalities that were impossible to simulate in the standard model, but possible to be simulation secure using random oracle. Therefore we cannot always remove the random oracle and must develop a more nuanced approach: we need to build a specific adversary for which simulation does not work.

In a recent work [27], Mahmoody et al. show that there is no fully black-box construction of indistinguishability obfuscation (iO) from any primitive implied by a random oracle in a black-box way. In light of recent FE to iO transformations [3, 10], one might wonder if this rules out FE schemes in the RO model. However, these transformations are non-black box.

High level description of impossibility Recall that we want to design an adversary $\mathcal{A}^\star$ that will build a list of "high frequency" random oracle queries and responses associated with each secret key and the challenge ciphertext. It will use this list later in the decryption phase to "cut-off" the random oracle at an appropriate time.

$\mathcal{A}^\star$ starts off by querying the key-generation oracle at random points $x_1, \ldots, x_q$ in the domain of wPRF, and gets $\mathsf{sk}_1, \ldots, \mathsf{sk}_q$ in return. The RO queries made by the key-generation oracle are *hidden* from the adversary, so $\mathcal{A}^\star$ tries to find them by encrypting several randomly chosen seeds using the master public key, and then decrypting them with $\mathsf{sk}_1, \ldots, \mathsf{sk}_q$.[4] The RO queries made during the decryption process are recorded in a list Γ. The hope is that Γ will capture the RO queries that were made in generating a key sk_i.

Note that one cannot hope to capture *all* RO queries required for decryption: Suppose a polynomial number Y of high frequency queries associated with sk_i is collected, but there is an RO call that is made during key-generation which is used during $1/2Y$ fraction of the decryptions. Then it will be the case that with some non-negligible probability, Γ will fail to aid in the decryption of the challenge ciphertext with sk_i. Instead of trying to solve this issue, we make our analysis work with a decryption that might fail some of the time. For this purpose, we extend the incompressibility argument of Agrawal et al. to work even for *approximate* compression.

We are not quite done yet. Even though we have captured most of the hidden RO queries involved in key-generation that are also needed for decryption, we still need to capture those that are involved in the encryption of the challenge message, as they are also *hidden* and may be required during decryption.[5] Suppose $\mathcal{A}^\star$ outputs a randomly chosen seed $s^\star$ as the challenge message, and gets $\mathsf{ct}^\star$ in return. In order to find out RO queries associated with $\mathsf{ct}^\star$, $\mathcal{A}^\star$ cannot generate secret keys on its own (like in the pre-challenge phase when it generated ciphertexts); it must make-do with the secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_q$ that were issued earlier. Thus, the idea is to decrypt $\mathsf{ct}^\star$ with some fraction δ of the keys using RO, recording the queries in the list Γ. It then cuts off the random oracle, and decrypts $\mathsf{ct}^\star$ with the remaining keys using the list Γ. If a query is not found in Γ, then a random value is used for it (as well as recorded in Γ for consistent responses in future). The adversary outputs 1 if a large fraction of these decryptions are correct; that is, if the decryption of $\mathsf{ct}^\star$ using sk_i outputs $\mathsf{wPRF}(s^\star, x_i)$.

In the real world, as we will see, the adversary outputs 1 with noticeable probability. On the other hand, we show that in the ideal world, the adversary outputs 1 only with negligible probability. For the adversary to output 1 in the ideal world, the simulator needs to somehow program the ciphertext and

[4] It is important that this is done before the challenge message is put out, otherwise simulator will get an opportunity to program in additional information through the random oracle.

[5] The RO queries made while setting up the FE system are also hidden from the adversary, but we ignore them here for simplicity.

the post-challenge random oracle queries so that a large number of decryptions succeed. The only opportunity a simulator has of programming post-challenge RO responses is when δ fraction of the keys are used for decrypting $\mathsf{ct}^\star$. By choosing δ appropriately, we can ensure that the simulator is not able to program the RO queries to the extent that most of the remaining decryptions succeed.

Looking back. A simulator's success in the RO model depends on when it comes to know what to program and how much can it program. When dealing with the attacker $\mathcal{A}^\star$ described above, it gets a large amount of information, $\mathsf{wPRF}(s^\star, x_1), \ldots, \mathsf{wPRF}(s^\star, x_q)$, only in the challenge phase. Since all the key queries come before that, programming the secret keys is ruled out. If there was no random oracle, then the only possible avenue to program is the challenge ciphertext, but AGVW shows that it is not possible to compress so much information into a small ciphertext. Now with the random oracle, it might have been possible to program this information *if* there were many RO queries after the challenge phase. However, our adversary makes only a bounded number of post-challenge RO queries, and as a result, it is not possible to program all of $\{\mathsf{wPRF}(s^\star, x_i)\}$ in these RO responses.

An alternative approach to proving impossibility Concurrent to our work, Bitansky, Lin and Paneth [7] showed an alternate approach for removing random oracles. Unlike our current impossibility, their approach requires multiple ciphertexts. We sketch the main ideas here.

This approach uses a notion of obfuscation called 'exponentially-efficient obfuscation', introduced by Lin et al. [26]. An exponentially-efficient obfuscator is allowed to run in subexponential time, and the obfuscated program is also allowed to be subexponential in the input length. For security, Lin et al. considered the $i\mathcal{O}$ equivalent, where the obfuscation of two functionally identical programs should be computationally indistinguishable. However, one can even consider simulation based notions where the output of the obfuscator can be simulated by a simulator having only black box access to the program.

In a recent work, Bitansky et al. [8] showed that IND-secure functional encryption can be used to construct exponentially-efficient indistinguishability obfuscation [26] in a black-box manner. While there exist other transformations from FE to obfuscation [3, 10], the BNPW transformation is the only known black-box transformation, and this is important when studying FE or obfuscation in the random oracle model. Using the BNPW transformation, one can argue that simulation secure FE in the random oracle model implies simulation-secure exponentially-efficient obfuscation in the random oracle model. Therefore, to rule out FE in the random oracle model, it suffices to show that there exist certain functionalities for which we cannot obtain simulation-secure exponentially-efficient obfuscation in the random oracle model.

This can be achieved using the techniques of Canetti et al. [15], who showed an impossibility result for VBB obfuscation in the random oracle model. Canetti et al. showed that if there exists a VBB obfuscator in the random oracle model,

then there exists an 'approximate' VBB obfuscator in the standard model. A similar argument can be used to show that if there exists simulation-secure exponentially efficient obfuscation in the random oracle model, then there exists approximately correct simulation-secure exponentially-efficient obfuscator in the standard model.

Finally, one needs to show that it is impossible to construct approximately correct simulation-secure exponentially-efficient obfuscators for certain function classes. This argument is similar to the incompressibility argument that we use. Let C be a circuit that performs PRF evaluation, and consider the obfuscation of C. A simulator must output an obfuscation given only black box access to the PRF function, which in turn is indistinguishable from a truly random function. Therefore, the simulator must output a subexponential sized string that approximately explains a truly random function, which is impossible.

1.2 A New Possibility Result in the Random Oracle Model

Now that we know that simulation security is impossible for unbounded queries even in the random oracle model, we turn to asking whether this model can be leveraged to support simulation security in any situations where it is impossible in the standard model. We already have one such example from the work of Boneh et al. [12] which gives both a standard model impossibility and a random oracle feasibility result for public index schemes. Thus, we are interested in new examples that go beyond the public index class. In this paper, we show the following possibility result:

Theorem 2 (Possibility, informal). *There exists a simulation secure FE scheme for the class of all polynomial-depth circuits in the random oracle model secure against any poly-time attacker who makes an unbounded number of messages queries, but an a-priori bounded number of key queries, based on semantically-secure public-key encryption and pseudo-random generators computable by low-depth circuits.*

Recall that such a security notion cannot be achieved even for the simple functionality of IBE in the standard model [12].

One-bounded FE. Our starting point is a *one-bounded* simulation-secure FE scheme for all circuits, i.e., a scheme where the attacker can only make one key query, based just on the semantic security of public-key encryption. Our scheme is a variant of the Sahai-Seyalioglu [31]. Let $\mathcal{C}$ be a family of circuits wherein each circuit can be represented using t bits. Suppose U_x is a universal circuit that takes a $C \in \mathcal{C}$ as input, and outputs $C(x)$. The set-up algorithm of our FE scheme generates $2t$ key pairs of a semantically-secure public-key encryption scheme. The $2t$ public keys $(\mathsf{pk}_{1,0}, \mathsf{pk}_{1,1}), \ldots, (\mathsf{pk}_{t,0}, \mathsf{pk}_{t,1})$ form the master public key, and the $2t$ private keys $(\mathsf{sk}_{1,0}, \mathsf{sk}_{1,1}) \ldots, (\mathsf{sk}_{t,0}, \mathsf{sk}_{t,1})$ are kept secret. In order to encrypt a message x, a garbled circuit for U_x is generated. Suppose $w_{i,b}$ for $i = 1, \ldots, t$ and $b = 0, 1$ are the *wire-labels* of U_x for its t input bits. Then the $(i, b)^{th}$ component of the ciphertext consists of two parts: an encryption of

a random value $r_{i,b}$ under $\mathsf{pk}_{i,b}$, and $w_{i,b}$ blinded with the hash of $r_{i,b}$. The key for a circuit C represented using bits $\beta_1, \ldots, \beta_t$ is simply the private keys corresponding to those bits, i.e., $\mathsf{sk}_{\beta_1}, \ldots, \mathsf{sk}_{\beta_t}$.

It is easy to see that the one-bounded FE scheme is correct. Specifically, the secret key for C will allow one to recover r_{i,β_i} for $i = 1, \ldots, t$. Then by running the hash function on these values, the w_{i,β_i} can be unblinded and used to evaluate the garbled circuit.

Let us now see how a simulator $\mathcal{S}$ can generate ciphertexts and a key from the right distribution in the ideal world. If the only allowed key query is made before the challenge phase for a circuit C, then $\mathcal{S}$ just runs the normal key generation algorithm, and later when adversary outputs a challenge message $x^\star$, it can generate a garbled circuit using just $C(x^\star)$.[6] When the adversary's key query is after the challenge message, however, $\mathcal{S}$ does not get any information in the challenge phase. In particular, it does not know which universal circuit to garble. Here the random oracle allows the simulator to *defer* making a decision until after the key query is made. It can set the second part of the (i, b)th ciphertext component to be a random number $z_{i,b}$ because, intuitively, adversary does not know $r_{i,b}$ (it is encrypted) so a hash of it is completely random. When adversary queries with a circuit C afterwards, simulator can program the random oracle's response on $r_{i,b}$ to be $z_{i,b} \oplus w_{i,b}$, so that decryption works out properly.

Bounded collusion FE. Using the one-bounded scheme in a black-box way, we can design an FE scheme secure against any a-priori bounded collusions for the class $\mathsf{NC1}$, using Gorbunov et al.'s [20] transformation. While their transformation was proved secure for only one challenge message, we show that the same ideas also work for unbounded number of challenge messages. If the underlying one-bounded scheme is secure against any number of challenge messages, then so is the scheme obtained after applying their transformation.

Related work. Sahai and Seyalioglu [31] were the first to use randomized encodings to design an FE system. Their scheme can issue one key *non-adaptively* for any function. Our one-bounded scheme can be seen as an extension of theirs to additionally support post-challenge key query. The random oracle allows a simulator to not commit to any value in the ciphertext until the function evaluation is made available.

Goldwasser et al. [19] also designed an FE system that can issue one pre-challenge key. Their scheme has succinct ciphertexts (independent of circuit size) but security is proved under stronger assumptions.

Iovino and Zebroski [24] present two results on simulation-secure FE in the public-key setting. First, they have a construction for a bounded number of challenge ciphertexts and pre-challenge key queries (and unbounded post-challenge queries), where key size grows with number of challenge ciphertexts but the ciphertext size is constant. The encryption/decryption time grows with the number

[6] In fact, if we just want pre-challenge key query security, then there is no need for random oracle.

of pre-challenge key queries. The second construction is for bounded key queries and challenge ciphertexts, but with constant size keys and ciphertexts. Here the encryption/decryption times depend on the bound on number of key queries and challenge ciphertexts. Both these results use extractability obfuscation. Our positive result presents a construction where the number of challenge ciphertexts is unbounded, but key queries are bounded. Therefore, our positive result and the results of Iovino and Zebroski are incomparable. Moreover, our construction only requires PKE and low-depth PRGs, whereas their constructions require stronger assumptions.

1.3 Another Impossibility Result

A natural question to ask is whether we can construct a simulation secure FE scheme in the random oracle model that can handle unbounded ciphertext queries, followed by an unbounded number of post-challenge key queries. We show that this is also impossible, assuming the existence of weak pseudorandom functions.

Theorem 3. *There does not exist a* $(0, \mathsf{poly}, \mathsf{poly})$*-simulation secure FE scheme for the class of (weak) pseudo-random functions in the random oracle model.*

Once again we interpret messages as seeds to a weak PRF family wPRF and secret keys as points in the domain of the PRF. A very different way to attack an FE scheme is needed though because no key query can be made before the challenge phase.

The new attacker $\mathcal{A}^\star$ starts off by outputting randomly chosen seeds $s_1, \ldots, s_k$ for wPRF, and gets ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$ in return. The RO queries made in the encryption process are *hidden* from $\mathcal{A}^\star$, and it might need some of them later during decryption. So, it requests secret keys for randomly chosen points $x_1, \ldots, x_q$, and gets $\mathsf{sk}_1, \ldots, \mathsf{sk}_q$ in return. Then it decrypts every ct_i with sk_j and records the RO queries made in a list Γ. An important point to note here is that the simulator gets some information about the seeds chosen earlier when key-queries are made. Specifically, it gets $\mathsf{wPRF}(s_1, x_j), \ldots, \mathsf{wPRF}(s_k, x_j)$ when x_j is the query.

$\mathcal{A}^\star$ now picks a random point x^* and requests a secret key for it. The goal is to use the key obtained, say sk^*, to decrypt the challenge ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$ later. But, in order to do so, $\mathcal{A}^\star$ also needs to find out the RO queries made during key-generation that may also be required for decryption. To solve this problem, we use the same idea as in the previous impossibility result: encrypt some random seeds on your own and decrypt them with sk^*, while adding the RO queries made to Γ.

Finally, $\mathcal{A}^\star$ decrypts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$ with sk^* *without* invoking the random oracle, using the list Γ instead. In the real world, at least a constant fraction of the decryptions succeed. The analysis is similar to that of the previous impossibility result, but with the role of ciphertext and key reversed. The ideal world analysis, on the other hand, need more care because of two reasons. First, as pointed out earlier, some information about the seeds $s_1, \ldots, s_k$ is leaked when

post-challenge key queries are made. Second, the simulator needs to compress the evaluation of wPRF on seeds $s_1, \ldots, s_k$ and a common point x^*, instead of one seed and multiple points as in the $(\mathsf{poly}, 1, 0)$ impossibility. At the same time, however, the only opportunity a simulator has of programming RO responses after learning $\mathsf{wPRF}(s_1, x^*), \ldots, \mathsf{wPRF}(s_k, x^*)$ is when ciphertexts for random seeds are decrypted with sk^* with the help of RO. So, it is conceivable that one can exploit the security of wPRF to argue that it is impossible to compress $\mathsf{wPRF}(s_1, x^*), \ldots, \mathsf{wPRF}(s_k, x^*)$ into a small key and a small number of RO responses. We show that this is indeed the case in the full version [1].

1.4 Relation to De Caro et al. and Functional Encryption for Circuits with Random Oracle Gates

At the time of the initial posting of our work, De Caro et al. [16] stated (Theorem 3) that indistinguishability security for FE schemes in the random oracle model implied simulation security, resulting in an apparent discrepancy with our results. After our work was posted we contacted the authors to point out this dissonance. The authors informed us that they had earlier become aware of an issue with the theorem statement, but had not yet prepared an update to their posting. They stated that they intended to update it to a statement that indistinguishability-based definition of "functional encryption for circuits with random oracle gates" implied simulation security.

At the time, the notion of functional encryption for circuits with random oracle gates had not previously appeared in the literature and we were unable to deduce the intended definition from the phrase. Subsequently, the authors provided a revision which defined the concept and provided a transformation in the random oracle model which showed this new notion implies (regular) simulation security [17]. However, since our work shows such a notion is impossible to achieve, this must imply that this indistinguishability notion of "functional encryption for circuits with random oracle gates" was impossible to realize to begin with.

Despite sharing the term *random oracle* the new concept proposed in their revision is quite different than how the random oracle model was proposed [6]. Recall, that a cryptographic system built in the random oracle model will have the same algorithms and definitions as the standard model counterpart with the exception that each algorithm is allowed oracle access to a random function. We emphasize that the random oracle model in of itself is not impossible, it is just simply a different model.[7] Prior works would typically first establish provable security in the random oracle model and then apply the heuristic of replacing the random oracle calls with those to a hash function. It is this last step where security can actually be lost; in some cases no matter what the hash function

[7] We note that in practice one could actually instantiate this model with a trusted third party that dynamically builds a random table. However, this is not done since presumably one does not want to require online communication and introduce such a trusted third party.

is [14]. The concept of IND-FE in the random oracle model is not impossible to achieve (as far as we know), but we show that it is still insufficient to get simulation security. This impossibility holds for the random oracle model itself and is completely independent of the hash function replacement heuristic.

In the concept of functional encryption with random oracle gates as defined in the revision of [17] the random oracle is not just used as a tool to help augment functional encryption, but actually incorporated into a definition of functional encryption as the descriptions of a functionality f will depend on the random oracle. (Due to space limitations we refer the reader to [17] for a detailed description of the new definition.) As a simple argument will show, this new indistinguishability notion, unlike standard FE in the random oracle model, is impossible to begin with. So the addition of random oracle gates to FE circuits moves one to a primitive that is unachievable.

The combination of our simulation impossibility results with the implications from [17] imply this new notion of indistinguishability FE with random oracle gates is impossible to achieve. However, there is a much simpler and direct argument, which we provide in the full version of our paper [1].

1.5 Interpreting our Impossibility Results

Impossibility results for simulation secure functional encryption in the standard model were already known before our work. If we take any FE system secure in the Random Oracle Model and then take the heuristic of replacing the oracle calls with some hash function family, then we have a standard model FE scheme. We know this new system to be impossible to be simulation secure from prior work. So a natural question to ask is what new interpretations does our result provide. We believe there are two main points here.

First, an interpretation of our result is understanding FE in idealized models. While the random oracle model is closely associated with the random oracle heuristic (i.e. replacing oracle calls with hash functions), there are different possible ways to try to "instantiate" a cryptosystem described in the random oracle model. One possibility is to replace calls to the random oracle with secure hardware tokens. Another could be a use of a blockchain.

In addition, in the interest of getting a better and deeper scientific understanding it is useful to map out cryptography in both the standard and random oracle models. There has been precedent for this in our community. For example, the Boneh et al. [12] paper which gave some examples of schemes (simulation secure FE schemes where the adversary sends unbounded challenge messages, followed by one key query) that were possible in the random oracle model, but impossible otherwise. Going further out, to best understand non-committing encryption it is useful to know both that it is impossible in the standard model and that it is possible in the RO model.

Secondly, we also posit that there may be some forms of security that lie in between simulation security and indistinguishability security, but that are hard for us to understand or formally define. Suppose there did exist an FE scheme that was simulation secure in the RO model, and one did apply the random

oracle heuristic to it. It is possible that even if this new scheme is not simulation secure, the transformation could result in some gain of security. Perhaps this gain in security might even be what is right or needed for a particular application. One example is that while the Fiat-Shamir heuristic applied to zero knowledge protocol does not give a simulation secure NIZK, but might give the right form of security needed for a particular application (e.g. its use in some cryptocurrency).

2 Preliminaries

We use λ to denote the security parameter. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. If A is an algorithm, then $a \leftarrow A(\cdot)$ or $A(\cdot) \rightarrow a$ denote that a is the output of running A on the specified inputs. If $\mathcal{D}$ is a distribution, then $s \leftarrow \mathcal{D}$ denotes that s is a sample drawn according to it. Also, $x \xleftarrow{R} X$ denotes drawing a value x uniformly at random from the set X.

For two distribution ensembles $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, we use $\mathcal{X} \overset{c}{\approx} \mathcal{Y}$ to denote that $\mathcal{X}$ is computationally indistinguishable from $\mathcal{Y}$. Lastly, for two vectors $u = (u_1, \ldots, u_n)$ and $v = (v_1, \ldots, v_n)$, their Hamming distance $\mathsf{HD}(u, v)$ is defined to be the number of points where they don't match, i.e., the size of set $\{i \in [n] \,|\, u_i \neq v_i\}$.

2.1 Weak Pseudo-random Functions

Our impossibility results rely on the existence of circuit families whose output cannot be *compressed* by a significant amount. In Section 4, we will show that a specific circuit family built from pseudo-random functions (PRFs) is not compressible. In fact, like Gorbunov et al. [20], a weaker type of PRF where adversary only gets evaluation at random points suffices for our purpose.

Definition 1 (Weak PRFs). *Let n, m, p be polynomials in λ. Let $\mathsf{wPRF} = \{\mathsf{wPRF}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of efficiently computable functions such that $\mathsf{wPRF}_\lambda : \{0,1\}^{n(\lambda)} \times \{0,1\}^{m(\lambda)} \rightarrow \{0,1\}^{p(\lambda)}$, where the first input is called the seed. Pick a seed $s \xleftarrow{R} \{0,1\}^{n(\lambda)}$ and $\ell + 1$ points $x_1, \ldots, x_\ell, x^\star \xleftarrow{R} \{0,1\}^{m(\lambda)}$. Let D_ℓ be the ℓ-tuple of values $(x_1, \mathsf{wPRF}_\lambda(s, x_1)), \ldots, (x_\ell, \mathsf{wPRF}_\lambda(s, x_\ell))$. Then the wPRF family is a weak pseudo-random function family if for every ℓ polynomial in λ,*

$$\{D_\ell, x^\star, \mathsf{wPRF}_\lambda(s, x^\star)\}_{\lambda\in\mathbb{N}} \quad \overset{c}{\approx} \quad \{D_\ell, x^\star, r\}_{\lambda\in\mathbb{N}},$$

where r is a random string of length $p(\lambda)$.

Below we present two alternate definitions of security for a weak pseudorandom family. The first one is a standard definition for PRFs/weak PRFs, while the second one is introduced for our final impossibility result. They both follow from Definition 1 above through simple hybrid arguments.

Definition 2 (Weak PRFs, many points). *Let $\mathsf{wPRF} = \{\mathsf{wPRF}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family as in Definition 1. Pick $s \xleftarrow{R} \{0,1\}^{n(\lambda)}$, $x_1, \ldots, x_\ell \xleftarrow{R} \{0,1\}^{m(\lambda)}$, and*

$r_1, \ldots, r_\ell \xleftarrow{R} \{0,1\}^{p(\lambda)}$. *Then the* wPRF *family is a weak PRF family for many points if for every* ℓ *polynomial in* λ*,*

$$\{(x_1, \mathsf{wPRF}_\lambda(s, x_1)), \ldots, (x_\ell, \mathsf{wPRF}_\lambda(s, x_\ell))\}_{\lambda \in \mathbb{N}} \quad \overset{c}{\approx} \quad \{(x_1, r_1), \ldots, (x_\ell, r_\ell)_{\lambda \in \mathbb{N}}.$$

Definition 3 (Weak PRFs, many seeds with aux). *Let* $\mathsf{wPRF} = \{\mathsf{wPRF}_\lambda\}_{\lambda \in \mathbb{N}}$ *be a family as in Definition 1. Pick* k *seeds* $s_1, \ldots, s_k \xleftarrow{R} \{0,1\}^{n(\lambda)}$ *and* $\ell+1$ *points* $x_1, \ldots, x_\ell, x^\star \xleftarrow{R} \{0,1\}^{m(\lambda)}$*. Let* $D_{k,\ell}$ *be the* $k \cdot \ell$*-tuple of values* $(x_1, \mathsf{wPRF}_\lambda(s_1, x_1))$, $\ldots$, $(x_\ell, \mathsf{wPRF}_\lambda(s_1, x_\ell))$, $\ldots$, $(x_1, \mathsf{wPRF}_\lambda(s_k, x_1)), \ldots, (x_\ell, \mathsf{wPRF}_\lambda(s_k, x_\ell))$*. Then the* wPRF *family is a weak PRF family for many seeds with auxiliary information if for every* k, ℓ *polynomial in* λ*,*

$$\{D_{k,\ell}, x^\star, \mathsf{wPRF}_\lambda(s_1, x^\star), \ldots, \mathsf{wPRF}_\lambda(s_k, x^\star)\}_{\lambda \in \mathbb{N}} \quad \overset{c}{\approx} \quad \{D_{k,\ell}, x^\star, r_1, \ldots, r_k\}_{\lambda \in \mathbb{N}},$$

where $r_1, \ldots, r_k$ *are random strings of length* $p(\lambda)$*.*

2.2 Randomized Encodings

We use decomposable randomized encodings [20] to simplify the description of our FE schemes. They are known to exist for all circuits due to the works of [34, 4].

Definition 4 (Randomized Encodings). *Let* $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ *be a family of circuits, where each circuit* $C \in \mathcal{C}_\lambda$ *takes an* $n(\lambda)$ *bit input and produces an* $m(\lambda)$ *bit output. A* decomposable randomized encoding RE *of* $\mathcal{C}$ *consists of two* PPT *algorithms:*

- $\mathsf{RE.Encode}(1^\lambda, C)$ *: It takes a circuit* $C \in \mathcal{C}_\lambda$ *as input, and outputs a randomized encoding* $((w_{1,0}, w_{1,1}), \ldots, (w_{n(\lambda),0}, w_{n(\lambda),1}))$*.*
- $\mathsf{RE.Decode}(1^\lambda, (\tilde{w}_1, \ldots, \tilde{w}_{n(\lambda)}))$ *: It takes an encoding* $(\tilde{w}_1, \ldots, \tilde{w}_{n(\lambda)})$ *and outputs* $y \in \{0,1\}^{m(\lambda)} \cup \{\perp\}$*.*

Correctness Let $C \in \mathcal{C}_\lambda$ *be any circuit, and let* $((w_{1,0}, w_{1,1}), \ldots, (w_{n,0}, w_{n,1})) \leftarrow \mathsf{RE.Encode}(1^\lambda, C)$*. For any input* $x \in \{0,1\}^{n(\lambda)}$*,* $\mathsf{RE.Decode}(1^\lambda, (w_{1,x_1}, \ldots, w_{n(\lambda),x_{n(\lambda)}})) = C(x)$*.*

Security To define the security of such a scheme, consider the following two distributions:

- $\mathsf{Real}^{\mathsf{RE}}_{\mathcal{A}}(\lambda)$*. Run* $\mathcal{A}(1^\lambda)$ *to get a* $C \in \mathcal{C}_\lambda$ *and an* $x \in \{0,1\}^{n(\lambda)}$*. Then run* RE.Encode *on input* C *to get an encoding* $((w_{1,0}, w_{1,1}), \ldots, (w_{n(\lambda),0}, w_{n(\lambda),1}))$*. Output* $\{w_{i,x_i}\}_{i \in [n(\lambda)]}$*.*
- $\mathsf{Ideal}^{\mathsf{RE}}_{\mathcal{S}}(\lambda)$*. Run* $\mathcal{A}(1^\lambda)$ *to get a* $C \in \mathcal{C}_\lambda$ *and an* $x \in \{0,1\}^{n(\lambda)}$*. Output* $\mathcal{S}(1^\lambda, C, C(x))$*.*

A randomized encoding scheme RE *is* secure *if for every* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *simulator* $\mathcal{S}$ *such that*

$$\mathsf{Real}^{\mathsf{RE}}_{\mathcal{A}}(\lambda) \overset{c}{\approx} \mathsf{Ideal}^{\mathsf{RE}}_{\mathcal{S}}(\lambda).$$

3 Functional Encryption in the Random Oracle Model

A functional encryption scheme for a function space $\mathbb{F} = \{\mathbb{F}_\lambda\}_{\lambda\in\mathbb{N}}$ and a message space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ in the random oracle model consists of four PPT algorithms that have access to a random oracle $\mathcal{O} : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{m(\lambda)}$, where ℓ and m are polynomials. The algorithms are described as follows:

- $\mathsf{Setup}^{\mathcal{O}}(1^\lambda)$: It takes the security parameter (in unary representation) as input and outputs a public key pk and a master secret key msk.
- $\mathsf{KeyGen}^{\mathcal{O}}(\mathsf{msk}, f)$: It takes the master secret key msk and a circuit $f \in \mathbb{F}_\lambda$ as inputs, and outputs a secret key sk_f for the circuit.
- $\mathsf{Encrypt}^{\mathcal{O}}(\mathsf{pk}, x)$: It takes the public key pk and a value $x \in \mathcal{X}_\lambda$ as inputs, and outputs a ciphertext ct_x.
- $\mathsf{Decrypt}^{\mathcal{O}}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct})$: It takes the public key pk, a secret key sk, and a ciphertext ct as inputs, and outputs a value y or $\perp$.

Correctness. The four algorithms defined above must satisfy the following correctness property. For all values of the security parameter λ, for every $f \in \mathbb{F}_\lambda$ and $x \in \mathcal{X}_\lambda$, all random oracles $\mathcal{O}$, and all $(\mathsf{pk}, \mathsf{msk})$ output by $\mathsf{Setup}^{\mathcal{O}}(1^\lambda)$,

$$\mathsf{Decrypt}^{\mathcal{O}}(\mathsf{pk}, \mathsf{KeyGen}^{\mathcal{O}}(\mathsf{msk}, f), \mathsf{Encrypt}^{\mathcal{O}}(\mathsf{pk}, x)) = f(x).$$

Without loss of generality, we can assume $\mathsf{Decrypt}$ to be deterministic.

One could consider weaker notions of correctness where a negligible probability of error is allowed.

Statistical Correctness. For all values of the security parameter λ, for every $f \in \mathbb{F}_\lambda$ and $x \in \mathcal{X}_\lambda$, all random oracles $\mathcal{O}$,

$$\Pr\left[\mathsf{Decrypt}^{\mathcal{O}}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}) = f(x) : \begin{array}{c}(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}^{\mathcal{O}}(1^\lambda) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}^{\mathcal{O}}(\mathsf{msk}, f) \\ \mathsf{ct} \leftarrow \mathsf{Encrypt}^{\mathcal{O}}(\mathsf{pk}, x)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

3.1 Simulation-based Security

Definition 5 (Experiments). *Let* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be a functional encryption scheme. For any* PPT *algorithms* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *and* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$*, Figure 1 defines two experiments* $\mathsf{Real}^{\mathsf{FE}}_{\mathcal{A}}(\lambda)$ *and* $\mathsf{Ideal}^{\mathsf{FE}}_{\mathcal{A},\mathcal{S}}(\lambda)$*. In the figure,* q_c *denotes the length of challenge message vector* $\mathbf{x}$ *output by* $\mathcal{A}_1$ *and* q_1 *denotes the number of key generation queries made before that.*

Definition 6 (Admissibility). *An adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *is* $(q_{\mathsf{pre}}(\lambda), q_{\mathsf{chal}}(\lambda), q_{\mathsf{post}}(\lambda))$-admissible *if in any run of the experiments* $\mathsf{Real}_{\mathcal{A}}(1^\lambda)$ *and* $\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}(1^\lambda)$*,* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *make at most* $q_{\mathsf{pre}}(\lambda)$ *and* $q_{\mathsf{post}}(\lambda)$ *key generation queries, respectively, and* $\mathcal{A}_1$ *outputs at most* $q_{\mathsf{chal}}(\lambda)$ *challenge messages.*

Experiment $\mathsf{Real}^{\mathsf{FE}}_{\mathcal{A}}(\lambda)$:

1. $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}^{\mathcal{O}}(1^\lambda)$
2. $(\mathbf{x}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_1^{\mathsf{KeyGen\text{-}RO}_1(\mathsf{msk},\cdot,\cdot)}(\mathsf{pk})$
3. $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}^{\mathcal{O}}(\mathsf{mpk}, x_i)$ for $i \in [q_c]$
4. $\alpha \leftarrow \mathcal{A}_2^{\mathsf{KeyGen\text{-}RO}_2(\mathsf{msk},\cdot,\cdot)}(\{\mathsf{ct}_i\}_{i\in[q_c]}, \mathsf{st}_{\mathcal{A}})$
5. Output α

Experiment $\mathsf{Ideal}^{\mathsf{FE}}_{\mathcal{A},\mathcal{S}}(\lambda)$:

1. $(\mathsf{pk}, \mathsf{st}_1) \leftarrow \mathcal{S}_1(1^\lambda)$
2. $(\mathbf{x}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_1^{\mathsf{KeyGen\text{-}RO}_1(\mathsf{st}_1,\cdot,\cdot)}(\mathsf{pk})$
3. $(\{\mathsf{ct}_i\}_i, \mathsf{st}_3) \leftarrow \mathcal{S}_3(\mathsf{st}_2, \{f_j(x_i)\}_{i,j})$ where $f_1, \ldots, f_{q_1}$ are key queries made by $\mathcal{A}_1$
4. $\alpha \leftarrow \mathcal{A}_2^{\mathsf{KeyGen\text{-}RO}_2(\mathsf{st}_3,\cdot,\cdot)}(\{\mathsf{ct}_i\}_{i\in[q_c]}, \mathsf{st}_{\mathcal{A}})$
5. Output α

In the Real-world experiment, the setup algorithm, using the random oracle $\mathcal{O}$, outputs public key pk and master secret key msk. The adversary $\mathcal{A}_1$ gets pk and has oracle access to $\mathsf{KeyGen\text{-}RO}_1$. This oracle responds to random oracle queries and key generation queries. It has msk hardwired and takes two inputs inp_1 and inp_2, where inp_1 specifies whether the query is a key generation query or a random oracle query. In the former case, $\mathsf{KeyGen\text{-}RO}_1$ outputs $\mathsf{KeyGen}^{\mathcal{O}}(\mathsf{msk}, \mathsf{inp}_2)$, while in the latter case, it outputs $\mathcal{O}(\mathsf{inp}_2)$. After polynomially many oracle queries to $\mathsf{KeyGen\text{-}RO}_1$, $\mathcal{A}_1$ outputs a vector $\mathbf{x}$ of ciphertext queries and a state $\mathsf{st}_{\mathcal{A}}$. The adversary $\mathcal{A}_2$ gets encryptions of all elements in $\mathbf{x}$ (note that x_i denotes the i^{th} entry in $\mathbf{x}$) and the state $\mathsf{st}_{\mathcal{A}}$. It also has oracle access to $\mathsf{KeyGen\text{-}RO}_2$, which is identical to $\mathsf{KeyGen\text{-}RO}_1$. After making polynomially many oracle queries, $\mathcal{A}_2$ outputs α.

In the Ideal-world experiment, the simulator $\mathcal{S}_1$ first computes the public key pk, and simulator state st_1. The adversary $\mathcal{A}_1$ gets pk and oracle access to $\mathsf{KeyGen\text{-}RO}_1$, which is simulator $\mathcal{S}_2$ in the ideal-world. The simulator $\mathcal{S}_2$ is stateful. It maintains an internal state st_2, gets $\mathcal{S}_1$'s state st_1 and takes tuple inputs $(\mathsf{inp}_1, \mathsf{inp}_2)$, which indicate whether it is a key generation query or a random oracle query. After polynomially many queries, adversary $\mathcal{A}_1$ outputs $\mathbf{x}$ and state $\mathsf{st}_{\mathcal{A}}$. The simulator $\mathcal{S}_3$ must give out encryptions of $\mathbf{x}$, using $\mathcal{S}_2's$ final state st_2 and $\{f_j(x_i)\}_{i\in[q_c],j\in[q_1]}$. The simulator outputs the ciphertexts as well as state st_3. Adversary $\mathcal{A}_2$ gets these ciphertexts, state $\mathsf{st}_{\mathcal{A}}$ and oracle access to $\mathsf{KeyGen\text{-}RO}_2$. In the ideal world, this oracle is $\mathcal{S}_4^{\mathsf{KeyIdeal}(\cdot)}(\mathsf{st}_3, \cdot, \cdot)$. Here, $\mathsf{KeyIdeal}$ takes as input a function f and outputs $(f(x_1), \ldots, f(x_{q_c}))$. Also, simulator $\mathcal{S}_4$ is stateful and has an internal state st_4. Finally, after polynomially many queries, $\mathcal{A}$ outputs α.

Fig. 1: Real and ideal experiments.

An adversary $\mathcal{A}$ *is* $(\mathsf{poly}, \mathsf{q}_{\mathsf{chal}}(\lambda), \mathsf{q}_{\mathsf{post}}(\lambda))$-admissible *if in any run of the experiments* $\mathsf{Real}_{\mathcal{A}}(1^\lambda)$ *and* $\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}(1^\lambda)$*,* $\mathcal{A}_1$ *is allowed to make an unbounded (but polynomial) number of pre-challenge key queries,* $\mathcal{A}_2$ *makes at most* $q_{\mathsf{post}}(\lambda)$ *key generation queries, and* $\mathcal{A}_1$ *outputs at most* $q_{\mathsf{chal}}(\lambda)$ *challenge messages. We can similarly define admissible adversaries where the number of challenge messages/post challenge key queries are unbounded.*

On the other hand, a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ *is* admissible *if whenever* $\mathcal{A}_2$ *makes a key query* f*,* $\mathcal{S}_4$ *queries* $\mathsf{KeyIdeal}$ *on* f *only.*

Definition 7 (Simulation security). *A functional encryption scheme* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is* $(q_{\mathsf{pre}}(\lambda), q_{\mathsf{chal}}(\lambda), q_{\mathsf{post}}(\lambda))$-$\mathsf{Sim}$*-secure for some polynomials* q_{pre}*,* q_{chal}*, and* q_{post}*, if there exists an admissible* PPT *simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ *such that for all* $(q_{\mathsf{pre}}(\lambda), q_{\mathsf{chal}}(\lambda), q_{\mathsf{post}}(\lambda))$*-admissible* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,*

$$\{\mathsf{Real}^{\mathsf{FE}}_{\mathcal{A}}(\lambda)\}_{\lambda\in\mathbb{N}} \stackrel{c}{\approx} \{\mathsf{Ideal}^{\mathsf{FE}}_{\mathcal{A},\mathcal{S}}(\lambda)\}_{\lambda\in\mathbb{N}}.$$

We also consider adversaries that make an unbounded (but polynomial) number of pre-challenge key queries/challenge messages/post-challenge key queries.

Definition 8 (Simulation security, unbounded queries). *A functional encryption scheme* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is* $(\mathsf{poly}, \mathsf{q}_{\mathsf{chal}}(\lambda), \mathsf{q}_{\mathsf{post}}(\lambda))$-$\mathsf{Sim}$*-secure for some polynomials* q_{chal}*, and* q_{post}*, if there exists an admissible* PPT *simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ *such that for all* $(\mathsf{poly}, \mathsf{q}_{\mathsf{chal}}(\lambda), \mathsf{q}_{\mathsf{post}}(\lambda))$*-admissible* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,*

$$\{\mathsf{Real}^{\mathsf{FE}}_{\mathcal{A}}(\lambda)\}_{\lambda\in\mathbb{N}} \stackrel{c}{\approx} \{\mathsf{Ideal}^{\mathsf{FE}}_{\mathcal{A},\mathcal{S}}(\lambda)\}_{\lambda\in\mathbb{N}}.$$

We can similarly define simulation security when q_{chal} *and* q_{post} *are unbounded.*

Note that in the real world an adversary has explicit access to the random oracle. In the ideal world, both the key generation and random oracles are simulated by $\mathcal{S}$ throughout the experiment.

Discussion on previous definitions of Sim-secure FE There are a number of definitions of simulation secure functional encryption [12, 30, 5, 2]. While these definitions are similar in spirit, there are minor differences. For instance, in the security game of [12, 2], the adversary makes pre-challenge key queries, followed by a challenge phase (where it queries for ciphertexts), followed by a post-challenge key query phase. The definition of [5] is more general as it allows arbitrary interleaving of encryption and key-generation queries. We use the AGVW definition [2] in this work, although we believe our impossibility result can also be extended to work for the definitions in [5].

4 Hardness of Approximate Compression

In this section, we will first define the notion of approximate compression, and then show that there are certain circuit families which are hard to approximately

compress. This section closely follows the work of Agrawal et al. [2], who defined the notion of (exact) compressibility of circuit evaluations, and showed that there exist certain circuit families that are (exact) incompressible.

Definition 9. *Let ℓ, t be polynomials and ϵ a non-negligible function. A class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ with domain $\mathcal{D} = \{\mathcal{D}_\lambda\}_\lambda$ and range $\mathcal{R} = \{\mathcal{R}_\lambda\}_\lambda$ is said to be (ℓ, t, ϵ)-approximately compressible if there exists a family of compression circuits $\mathsf{Cmp} = \{\mathsf{Cmp}_\lambda\}_\lambda$, a family of decompression circuits $\mathsf{DeCmp} = \{\mathsf{DeCmp}_\lambda\}_\lambda$, a polynomial poly, and a non-negligible function η, such that for all large enough λ the following properties hold:*

- *The circuits Cmp_λ and DeCmp_λ have size bounded by $\mathsf{poly}(\lambda)$.*
- *(compression) For all input $s \in \mathcal{D}_\lambda$ and circuits $C_1, C_2, \ldots, C_{\ell(\lambda)} \in \mathcal{C}_\lambda$,*

$$\left|\mathsf{Cmp}_\lambda\left(\{C_i, C_i(s)\}_{i\in[\ell(\lambda)]}\right)\right| \leq t(\lambda).$$

- *(approximate decompression) If s is chosen at random from $\mathcal{D}_\lambda$, $C_1, C_2, \ldots, C_{\ell(\lambda)}$ are chosen uniformly and independently from $\mathcal{C}_\lambda$, then*

$$\Pr\left[\mathsf{HD}\left(\mathsf{DeCmp}_\lambda\left(\{C_i\}_{i\in[\ell(\lambda)]}, \mathsf{Cmp}_\lambda\left(\{C_i, C_i(s)\}_{i\in[\ell(\lambda)]}\right)\right), \left(C_1(s), \ldots, C_{\ell(\lambda)}(s)\right)\right) \leq \epsilon(\lambda)\cdot t(\lambda)\right] \geq \eta(\lambda)$$

We will now show that weak PRFs can be used to construct a class of circuits that are not approximate compressible. We will then use the more general notion of approximate incompressibility, rather than the specific case of weak PRFs, in proving our impossibility results. For simplicity of presentation, in the lemma statement below, we use specific constants which will be sufficient for our main result. However, the lemma can be easily extended to work for general ℓ, t and ϵ. We assume that the weak PRF outputs a single bit.

Lemma 1. *Let $\mathsf{wPRF} = \{\mathsf{wPRF}_\lambda\}_\lambda$ be a family of weak pseudorandom functions (for many points), where $\mathsf{wPRF}_\lambda : \{0,1\}^{n(\lambda)} \times \{0,1\}^{m(\lambda)} \to \{0,1\}$. Consider the family of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$, where $\mathcal{C}_\lambda = \{\mathsf{wPRF}_\lambda(\cdot, x)\}_{x\in\{0,1\}^{m(\lambda)}}$. Let $t = t(\lambda)$ be any polynomial such that $t(\lambda) \geq \lambda$ for all $\lambda \in \mathbb{N}$. Then $\mathcal{C}$ is not $(16t, t, 1/8)$ approximate compressible.*

The proof of this lemma is given in the full version of our paper [1].

5 Impossibility of Simulation Secure FE

In this section we show that there does not exist a functional encryption scheme for the family of all polynomial-sized circuits that is $(\mathsf{poly}, 1, 0)$-Sim secure in the random oracle model. Specifically, we show that a simulation secure FE scheme cannot be constructed for any family of circuits that is not approximately

compressible (Definition 9). We exhibit an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that for *any* efficient simulator $\mathcal{S}$, the output of the real experiment, $\mathsf{Real}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda)$, is *distinguishable* from the output of the ideal experiment, $\mathsf{Ideal}^{\mathsf{FE}}_{\mathcal{A},\mathcal{S}}(1^\lambda)$ (Definition 8).

High level description of adversary. Let $\mathcal{C}$ be an approximate incompressible circuit family. The adversary $\mathcal{A}_1$ first asks for secret keys for a large number of randomly chosen circuits from C, and receives $\{\mathsf{sk}_1, \ldots, \mathsf{sk}_q\}$ in return. Next, it generates encryptions of many random messages. It then decrypts each of these ciphertexts using the q secret keys. The purpose of these encryptions followed by the decryptions is to capture the random oracle queries that would have occurred while computing the q secret keys, which may also be required when these keys are used again for decryption later. Let S_{keys} denote the set of random oracle queries that occur during these decryptions.

$\mathcal{A}_1$ chooses a random message x^*, and outputs it as the challenge (along with a state that consists of its view so far). $\mathcal{A}_2$ then receives a ciphertext ct^*. It decrypts ct^* using $\mathsf{sk}_1, \ldots, \mathsf{sk}_t$, for some small t. Let S_{ct^*} denote the set of random oracle queries during these t decryptions. The purpose of these t decryptions is to capture the random oracle queries that would have occurred during the encryption of x^*, which may also be required when ct^* is decrypted again in the next step.

Finally, $\mathcal{A}_2$ decrypts ct^* using the remaining $q - t$ secret keys. An important thing to note here is that $\mathcal{A}_2$ *turns off* the random oracle, and instead uses the queries that it has already recorded. If a new random oracle query is required, then it uses a randomly chosen string. It compares the decrypted values to the correct function evaluations, and outputs 1 if most decryptions are correct.

First, we show that in the real world, $\mathcal{A}_2$ outputs 1 with probability at least 3/4. Let us focus on one of the $q - t$ decryptions, using a secret key sk_j. At a high level, this decryption can go wrong if a random oracle query is made on z, and $z \notin S_{\text{keys}} \cup S_{\mathsf{ct}^*}$, but z was used during the computation of either sk_j or ct^*. We show that this event happens with low probability.

To complete the argument, we show that in the ideal world, $\mathcal{A}_2$ outputs 1 with probability around 1/2. In this world, the simulator receives q circuit evaluations on x^*, and must compress most of this information in the short challenge ciphertext and the random oracle queries made during the t post-challenge decryption operations. By choosing parameters carefully and appealing to the (approximate) incompressibility of the circuit family, we show that this is not possible.

5.1 Formal Description of Adversary

Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ be a family of circuits such that each circuit in $\mathcal{C}_\lambda$ takes an $n(\lambda)$-bit input and is not $(16t, t, 1/8)$ approximately compressible for all polynomials t such that $t(\lambda) \geq \lambda$. Let FE be a functional encryption scheme for this family in the random oracle model. We now formally define the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

Adversary $\mathcal{A}_1$. Let n_{key} and n_{enc} be polynomials in λ whose values will be fixed later. Let Γ be a list of (query, response) pairs that is empty at the beginning. $\mathcal{A}_1$ has four phases: setup, key query, random oracle query collection, and an output phase.

1. **Setup.** $\mathcal{A}_1$ receives the public key pk.
2. **Key query.** For $i \in [n_{\mathsf{key}}]$, it picks a circuit C_i at random from $\mathcal{C}_\lambda$, requests a secret key for C_i, and obtains sk_i in return.
3. **RO query collection 1.** $\mathcal{A}_1$ picks n_{enc} inputs $x_1, x_2, \ldots, x_{n_{\mathsf{enc}}} \xleftarrow{\mathrm{R}} \{0,1\}^{n(\lambda)}$. For $j \in [n_{\mathsf{enc}}]$, it runs $\mathsf{Encrypt}^{\mathcal{O}}(\mathsf{pk}, x_j)$ to obtain a ciphertext ct_j. The RO queries made during the encryption process are forwarded to the random oracle.
   Now each of the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_{n_{\mathsf{enc}}}$ are decrypted with key sk_i for every $i \in [n_{\mathsf{key}}]$. If an oracle query β is made by the $\mathsf{Decrypt}$ algorithm, $\mathcal{A}_1$ queries the random oracle with the same. The response, say γ, is given to the algorithm, and (β, γ) is added to Γ (if it is not already present).
4. **Output.** $\mathcal{A}_1$ picks an input $x^* \xleftarrow{\mathrm{R}} \{0,1\}^{n(\lambda)}$. It sets the state st to consist of pk, $C_1, \ldots, C_{n_{\mathsf{key}}}$, $\mathsf{sk}_1, \ldots, \mathsf{sk}_{n_{\mathsf{key}}}$, x^*, and Γ. Then it outputs (x^*, st).

Adversary $\mathcal{A}_2$. Let n_{eval} and n_{test} be polynomials in λ s.t. $n_{\mathsf{eval}}(\lambda) + n_{\mathsf{test}}(\lambda) = n_{\mathsf{key}}(\lambda)$ for all λ. (Their values will be fixed later.) $\mathcal{A}_2$ gets ct^* and st as input, and parses the latter to get pk, $C_1, \ldots, C_{n_{\mathsf{key}}}$, $\mathsf{sk}_1, \ldots, \mathsf{sk}_{n_{\mathsf{key}}}$, x^*, and Γ. $\mathcal{A}_2$ has three phases: random oracle query collection, test, and an output phase.

1. **RO query collection 2.** For every $i \in [n_{\mathsf{eval}}]$, ct^* is decrypted with sk_i. If an RO query β is made by the $\mathsf{Decrypt}$ algorithm, $\mathcal{A}_2$ queries the random oracle with the same. The response, say γ, is given to the algorithm, and (β, γ) is added to Γ (if it is not already present).
2. **Test.** In this phase, ct^* is decrypted with rest of the keys but *without* invoking the random oracle. In order to do so, a new list Δ is initialized first, then the following steps are executed for every $n_{\mathsf{eval}} + 1 \leq i \leq n_{\mathsf{eval}} + n_{\mathsf{test}}$. The decryption algorithm is run with inputs pk, sk_i, and ct^*. When it makes an RO query β, $\mathcal{A}_2$ checks whether there is an entry of the form (β, γ) in Γ or Δ (in that order) or not. If yes, then γ is given to $\mathsf{Decrypt}$ and it continues to run. Otherwise, a random bit-string γ' of length $m(\lambda)$ (the output length of the random oracle) is generated, (β, γ') is added to Δ, and γ' is given to $\mathsf{Decrypt}$. This process of providing responses to the RO queries of $\mathsf{Decrypt}$ continues till it terminates. Let out_i denote the output of $\mathsf{Decrypt}$, which could be $\perp$.
3. **Output.** For every $n_{\mathsf{eval}}+1 \leq i \leq n_{\mathsf{eval}}+n_{\mathsf{test}}$, check if out_i is equal to $C_i(x^*)$ (where x^* and C_i are part of the state transferred to $\mathcal{A}_2$). Let num be the number of keys for which this check succeeds. Output 1 if $\mathsf{num}/n_{\mathsf{test}} \geq 7/8$, else output 0.

To complete the description of $\mathcal{A}$, we need to define the polynomials n_{enc}, n_{eval} and n_{test} (recall that $n_{\mathsf{key}} = n_{\mathsf{eval}} + n_{\mathsf{test}}$). Let q_{Setup}, q_{Enc}, q_{KeyGen} and q_{Dec}

be upper-bounds on the number of RO queries made by Setup, $\mathsf{Encrypt}$, KeyGen and $\mathsf{Decrypt}$, respectively, as a function of λ. Also, let ℓ_{ct} be an upper-bound on the length of ciphertexts generated by $\mathsf{Encrypt}$. Then set

- $n_{\mathsf{enc}} = 4\lambda \cdot n_{\mathsf{key}} \cdot q_{\mathsf{KeyGen}}$,
- $n_{\mathsf{eval}} = 32\lambda\,(q_{\mathsf{Setup}} + q_{\mathsf{Enc}})$,
- $n_{\mathsf{test}} = 16(\ell_{\mathsf{ct}} + n_{\mathsf{eval}} \cdot q_{\mathsf{Dec}} \cdot m)$.

5.2 Real World Analysis

First, we will show that the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ described above outputs 1 with probability at least $3/4$ in the real world experiment, as long as the scheme FE is correct. To begin with, we classify the random oracle queries made during a run of $\mathcal{A}$ into different sets as follows:

- $\mathsf{S\text{-}RO}_{C_i}$ for $i \in [n_{\mathsf{key}}]$: random oracle queries made by KeyGen while generating secret key for C_i.
- $\mathsf{S\text{-}RO}_{\mathrm{keys}} = \bigcup_{i\in[n_{\mathsf{key}}]} \mathsf{S\text{-}RO}_{C_i}$: all random oracle queries during the key query phase of $\mathcal{A}_1$.
- $\mathsf{S\text{-}RO}_{x^*}$: random oracle queries made while encrypting x^* using pk.
- $\mathsf{S\text{-}RO}_{\mathrm{Dec}\text{-}i}$ for $i \in [n_{\mathsf{test}}]$: random oracle queries made during the decryption of ct^* using $\mathsf{sk}_{n_{\mathsf{eval}}+i}$.
- $\mathsf{S\text{-}RO}_{\Gamma\text{-}b}$: random oracle queries recorded during 'RO Collection Phase b' for $b \in \{1, 2\}$. Let $\mathsf{S\text{-}RO}_{\Gamma} = \mathsf{S\text{-}RO}_{\Gamma\text{-}1} \bigcup \mathsf{S\text{-}RO}_{\Gamma\text{-}2}$.
- $\mathsf{S\text{-}RO}_{\mathsf{Setup}}$: random oracle queries made during setup phase.

Lemma 2. *For any functional encryption scheme* FE *for the circuit family* $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$*, the adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *described in Section 5.1 outputs* 1 *in* $\mathsf{Real}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda)$ *with probability at least* $3/4 - \mathsf{negl}(\lambda)$.

Proof. We will use the correctness property of FE to prove this claim. We assume statistical correctness, i.e., for all random oracles $\mathcal{O} : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{m(\lambda)}$, $x \in \{0,1\}^{n(\lambda)}$, $C \in \mathcal{C}_\lambda$

$$\Pr\left[\mathsf{Decrypt}^{\mathcal{O}}\left(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}\right) = C(x) : \begin{array}{l}(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}^{\mathcal{O}}(1^\lambda) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}^{\mathcal{O}}\left(\mathsf{msk}, C\right) \\ \mathsf{ct} \leftarrow \mathsf{Encrypt}^{\mathcal{O}}\left(\mathsf{pk}, x\right)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

In particular, we do not assume the decryption to be deterministic.

Let Bad denote the event that the adversary outputs 0 at the end of the real world experiment. This event happens if at least 1/8th fraction of the n_{test} decryptions fail in the test phase. If $\mathsf{I\text{-}Dec}_i$ is an indicator variable that takes the value 1 in case the ith decryption *fails*, then Bad happens iff $\sum_{i\in[n_{\mathsf{test}}]} \mathsf{I\text{-}Dec}_i > 1/8 \cdot n_{\mathsf{test}}$. To analyze the probability of this event, we need to consider the random oracle queries required for decryption in the test phase. In this phase, $\mathcal{A}_2$ does not query the random oracle, but instead uses the list Γ. If some query

β is not present in Γ, then $\mathcal{A}_2$ tries to find it in Δ. If β is not found in Δ either, then a random value is chosen and recorded in Δ against β.

Now there are two ways in which the i^{th} decryption can fail. The first is if there is some entry (β, γ) in Δ such that β is also among the RO queries *hidden* from the adversary (and its response is not γ), i.e., the queries made during the setup phase, key query phase or challenge ciphertext generation. The second case is when the RO query responses are consistent, but the decryption is incorrect due to 'bad' decryption coins. The second failure happens with negligible probability (due to correctness of the FE scheme). In other words, the ith decryption succeeds with overwhelming probability if all the *needed* hidden RO responses are captured in either of the two RO collection phases. This is formalized in the following observation.

Observation 1 *Let* Bad-Dec *be the following event:*

$$\begin{gathered}\exists i \in [n_{\mathsf{test}}] \text{ s.t.} \\ (\mathsf{S\text{-}RO}_{\text{Dec-}i} \bigcap (\mathsf{S\text{-}RO}_{\mathsf{Setup}} \bigcup \mathsf{S\text{-}RO}_{\text{keys}} \bigcup \mathsf{S\text{-}RO}_{x^*}) \subseteq \mathsf{S\text{-}RO}_{\Gamma}) \\ \wedge \\ \mathcal{A}_2' s \text{ decryption of } \mathsf{ct}^* \text{ using } \mathsf{sk}_{n_{\mathsf{eval}}+i} \text{ does not output } C_{n_{\mathsf{eval}}+i}(x^*)\end{gathered}$$

There exists a negligible function $\mathsf{negl}(\cdot)$ *s.t.* $\Pr[\mathsf{Bad\text{-}Dec}] \leq \mathsf{negl}(\lambda)$ *where the probability is over the random coins used by setup, key generation, encryption, decryption and the adversary's choice of inputs.*

Proof. This observation follows from the statistical correctness of the scheme. Fix any index $i \in [n_{\mathsf{eval}}]$. Since $(\mathsf{S\text{-}RO}_{\text{Dec-}i} \bigcap (\mathsf{S\text{-}RO}_{\mathsf{Setup}} \bigcup \mathsf{S\text{-}RO}_{\text{keys}} \bigcup \mathsf{S\text{-}RO}_{x^*}) \subseteq \mathsf{S\text{-}RO}_{\Gamma})$, the oracle queries are consistent. Hence, we can use the correctness guarantee of the scheme to bound the probability of Bad-Dec.

Let $\mathsf{I\text{-}Dec\text{-}1}_i$ and $\mathsf{I\text{-}Dec\text{-}2}_i$ be indicator variables that are 1 iff $\mathsf{S\text{-}RO}_{\text{Dec-}i} \bigcap (\mathsf{S\text{-}RO}_{x^*} \bigcup \mathsf{S\text{-}RO}_{\mathsf{Setup}}) \not\subseteq \mathsf{S\text{-}RO}_{\Gamma}$ and $\mathsf{S\text{-}RO}_{\text{Dec-}i} \bigcap \mathsf{S\text{-}RO}_{\text{keys}} \not\subseteq \mathsf{S\text{-}RO}_{\Gamma}$, respectively. Then, $\mathsf{I\text{-}Dec}_i = 1$ iff either $\mathsf{I\text{-}Dec\text{-}1}_i = 1$ or $\mathsf{I\text{-}Dec\text{-}2}_i = 1$ (or both). Let Bad-1 and Bad-2 be events that happen iff $\sum_{i \in [n_{\mathsf{test}}]} \mathsf{I\text{-}Dec\text{-}1}_i > 1/16 \cdot n_{\mathsf{test}}$ and $\sum_{i \in [n_{\mathsf{test}}]} \mathsf{I\text{-}Dec\text{-}2}_i > 1/16 \cdot n_{\mathsf{test}}$, respectively. It is easy to see that

$$\Pr[\mathsf{Bad}] \leq \Pr[\mathsf{Bad\text{-}1}] + \Pr[\mathsf{Bad\text{-}2}] + \Pr[\mathsf{Bad\text{-}Dec}]$$

Below we show that $\Pr[\mathsf{Bad\text{-}1}] \leq \mathsf{negl}(\lambda)$ and $\Pr[\mathsf{Bad\text{-}2}] \leq 1/4$. Thus the lemma follows. ∎

Claim 1 $\Pr[\mathsf{Bad\text{-}1}] \leq \mathsf{negl}(\lambda)$.

Proof. Fix any random oracle $\mathcal{O}$, the randomness used in $\mathsf{Setup}^{\mathcal{O}}(1^{\lambda})$, challenge message x^*, and the randomness used in $\mathsf{Encrypt}^{\mathcal{O}}(\mathsf{pk}, x^*)$. This also fixes the sets $\mathsf{S\text{-}RO}_{\mathsf{Setup}}$ and $\mathsf{S\text{-}RO}_{x^*}$. Suppose a circuit C is picked at random from $\mathcal{C}_{\lambda}$, and a key, sk, is generated for it by running $\mathsf{KeyGen}^{\mathcal{O}}(\mathsf{msk}, C)$. For $z \in \mathsf{S\text{-}RO}_{\mathsf{Setup}} \cup \mathsf{S\text{-}RO}_{x^*}$, let ρ_z be the probability that z is an RO query in the decryption of ct^*

(the challenge ciphertext) with sk, where the probability is over the choice of C, the randomness used in KeyGen and the random coins used in decryption.

Let $X_{i,z}$ be an indicator variable that is 1 if an RO query on z is made during the ith decryption in post-challenge phase (either in the RO collection 2 or test phase). Note that the keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_{n_{\mathsf{key}}}$ are generated independently by choosing circuits $C_1, \ldots, C_{n_{\mathsf{key}}}$ uniformly at random, and the random coins used in each key generation and decryption are independently chosen. Thus for any z, the variables $X_{1,z}, \ldots, X_{n_{\mathsf{key}},z}$ are independent of each other, and $\Pr[X_{i,z} = 1] = \rho_z$ for every i.

We are interested in the probability that $\sum_{i \in [n_{\mathsf{test}}]} \mathsf{I\text{-}Dec\text{-}1}_i > n_{\mathsf{test}}/16$, i.e., in at least 1/16th fraction of the decryptions in the test phase, an RO query q is made s.t. q was also an RO query in either set-up or encryption of x^*, but it was not captured in either of the collection phases. Thus, there must exist a z s.t. $z \notin \mathsf{S\text{-}RO}_{\Gamma}$ (in particular, $z \notin \mathsf{S\text{-}RO}_{\Gamma\text{-}2}$) but an RO query on z is made in at least $n_{\mathsf{test}}/16|Q|$ of the decryptions, where $Q = \mathsf{S\text{-}RO}_{\mathsf{Setup}} \cup \mathsf{S\text{-}RO}_{x^*}$. (If $Q = \phi$ then Bad-1 cannot happen, and we are done.) Therefore,

$$\Pr\left[\sum_{i \in [n_{\mathsf{test}}]} \mathsf{I\text{-}Dec\text{-}1}_i > \frac{n_{\mathsf{test}}}{16}\right] \quad \leq \quad \sum_{z \in Q} \Pr\left[z \notin \mathsf{S\text{-}RO}_{\Gamma\text{-}2} \quad \wedge \quad \sum_{i \in [n_{\mathsf{test}}]} X_{i,z} > \frac{n_{\mathsf{test}}}{16|Q|}\right]$$

Based on the value of ρ_z, we can divide the rest of the analysis into two parts. Intuitively, if ρ_z is large, then the probability that z is not captured during RO collection phase is negligible. And when it is small, the probability that z causes too many decryptions to fail in the test phase is negligible. Since Q is polynomial in the security parameter, this will prove that the probability of Bad-1 is negligible as well. So now,

- If $\rho_z \geq 1/32|Q|$ then

$$\begin{aligned}\Pr[z \notin \mathsf{S\text{-}RO}_{\Gamma\text{-}2}] &= \Pr[X_{1,z} = 0 \wedge \ldots \wedge X_{n_{\mathsf{eval}},z} = 0] \\ &= \prod_{i \in [n_{\mathsf{eval}}]} \Pr[X_{i,z} = 0] \\ &= (1 - \rho_z)^{n_{\mathsf{eval}}} \leq e^{-n_{\mathsf{eval}}/32|Q|},\end{aligned}$$

  where the second equality follows from the independence of $X_{i,z}$. Recall that we set n_{eval} to be $32\lambda(q_{\mathsf{Setup}} + q_{\mathsf{Enc}})$, where q_{Setup} and q_{Enc} are upper-bounds on the number of RO queries made during Setup and $\mathsf{Encrypt}$, respectively. Thus, $e^{-n_{\mathsf{eval}}/32|Q|}$ is at most $e^{-\lambda}$.
- If $\rho_z < 1/32|Q|$ then expected value of $\sum_{i \in [n_{\mathsf{test}}]} X_{i,z}$ is at most $n_{\mathsf{test}}/32|Q|$. Using Chernoff bounds we can argue that,

$$\Pr\left[\sum_{i \in [n_{\mathsf{test}}]} X_{i,z} > \frac{n_{\mathsf{test}}}{16|Q|}\right] < e^{-\frac{1}{3} \cdot \frac{n_{\mathsf{test}}}{32|Q|}}.$$

  We know that $n_{\mathsf{test}} \geq n_{\mathsf{eval}}$. Thus, $e^{-\frac{1}{3} \cdot \frac{n_{\mathsf{test}}}{32|Q|}}$ is at most $e^{-\lambda}$ as well.

Claim 2 $\Pr[\mathsf{Bad}\text{-}2] \leq 1/4$.

Proof. Fix any random oracle $\mathcal{O}$, the randomness used in $\mathsf{Setup}^{\mathcal{O}}(1^\lambda)$, the circuits $C_1, \ldots, C_{n_{\mathsf{key}}}$ chosen in the key query phase, and the randomness used in $\mathsf{KeyGen}^{\mathcal{O}}(\mathsf{msk}, C_i)$ for $i \in [n_{\mathsf{key}}]$. This, in particular, fixes secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_{n_{\mathsf{key}}}$ and the set $\mathsf{S\text{-}RO}_{\mathrm{keys}}$. Consider the following experiment: $x \xleftarrow{\mathrm{R}} \{0,1\}^{n(\lambda)}$, $\mathsf{ct} \leftarrow \mathsf{Encrypt}^{\mathcal{O}}(\mathsf{pk}, x)$, and decrypt ct using sk_i for $i \in [n_{\mathsf{eval}}+1, n_{\mathsf{key}}]$. Let $\hat{\rho}_z$ be the probability that at least $n_{\mathsf{test}}/16|\hat{Q}|$ of the decryptions make an RO query on z, where $\hat{Q} = \mathsf{S\text{-}RO}_{\mathrm{keys}}$.

Let $Y_{j,z}$ be an indicator variable that is 1 iff an RO query on z is made in at least $n_{\mathsf{test}}/16|\hat{Q}|$ of the decryptions of ct_j with keys $\mathsf{sk}_{n_{\mathsf{eval}}+1}, \ldots, \mathsf{sk}_{n_{\mathsf{key}}}$ in the first phase of RO query collection. Note that the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_{n_{\mathsf{enc}}}$ are generated independently by choosing $x_1, \ldots, x_{n_{\mathsf{key}}}$ uniformly at random, and the decryption coins are also chosen independently for each decryption. Thus for any z, the variables $Y_{1,z}, \ldots, Y_{n_{\mathsf{enc}},z}$ are independent of each other, and $\Pr[Y_{j,z} = 1] = \hat{\rho}_z$ for every j. In a similar way, we can also define a random variable Y^*_z that indicates whether an RO query on z is made in at least $n_{\mathsf{test}}/16|\hat{Q}|$ of the decryptions of ct^* with keys $\mathsf{sk}_{n_{\mathsf{eval}}+1}, \ldots, \mathsf{sk}_{n_{\mathsf{key}}}$ in the test phase. Y^*_z is independent of $Y_{1,z}, \ldots, Y_{n_{\mathsf{enc}},z}$ and $\Pr[Y^*_z = 1] = \hat{\rho}_z$.

In a manner similar to the previous claim, we can argue that

$$\Pr\left[\sum_{i \in [n_{\mathsf{test}}]} \mathsf{I\text{-}Dec}\text{-}2_i > \frac{n_{\mathsf{test}}}{16}\right] \quad \leq \quad \sum_{z \in \hat{Q}} \Pr[z \notin \mathsf{S\text{-}RO}_{\Gamma\text{-}1} \wedge Y^*_z = 1]$$

If $z \notin \mathsf{S\text{-}RO}_{\Gamma\text{-}1}$, then none of the decryptions in the first phase of RO collection make a query on z. In particular, the variables $Y_{1,z}, \ldots, Y_{n_{\mathsf{enc}},z}$ are all zero in such a case. Therefore,

$$\begin{aligned}
\Pr[z \notin \mathsf{S\text{-}RO}_{\Gamma\text{-}1} \wedge Y^*_z = 1] &\leq \Pr[Y_{1,z} = 0 \wedge \ldots \wedge Y_{n_{\mathsf{enc}},z} = 0 \wedge Y^*_z = 1] \\
&= \Pr[Y^*_z = 1] \quad \cdot \prod_{j \in [n_{\mathsf{enc}}]} \Pr[Y_{j,z} = 0] \\
&= \hat{\rho}_z (1 - \hat{\rho}_z)^{n_{\mathsf{enc}}}
\end{aligned}$$

Once again we have two cases. If $\hat{\rho}_z \leq 1/4|\hat{Q}|$, then $\hat{\rho}_z(1-\hat{\rho}_z)^{n_{\mathsf{enc}}}$ is at most $1/4|\hat{Q}|$ as well. Otherwise, $(1-\hat{\rho}_z)^{n_{\mathsf{enc}}} \leq e^{-n_{\mathsf{enc}}/4|\hat{Q}|} \leq e^{-\lambda}$ because, recall that, n_{enc} is set to be $4\lambda \cdot n_{\mathsf{key}} \cdot q_{\mathsf{KeyGen}}$, where q_{KeyGen} is an upper-bound on the number of RO queries made during KeyGen. As a result, $\sum_{z \in \hat{Q}} \hat{\rho}_z (1-\hat{\rho}_z)^{n_{\mathsf{enc}}}$ is at most $1/4$.

5.3 Ideal world analysis

Next, we will show that any for PPT simulator, our adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ outputs 1 in the ideal world with negligible probability. Let t be a polynomial in λ such that $t = \ell_{\mathsf{ct}} + n_{\mathsf{eval}} \cdot q_{\mathsf{Dec}} \cdot m$ (so that $n_{\mathsf{test}} = 16t$) where, recall that, ℓ_{ct} is

the maximum length of any ciphertext generated by $\mathsf{Encrypt}$. Note that $q_{\mathsf{Dec}} \cdot m$ is the maximum number of bits obtained through the random oracle during any decryption, $n_{\mathsf{eval}} \cdot q_{\mathsf{Dec}} \cdot m$ is the maximum number of bits sent to the adversary during the second RO query collection phase, and $\ell_{\mathsf{ct}} + n_{\mathsf{eval}} \cdot q_{\mathsf{Dec}} \cdot m$ is the total number of bits the adversary receives after sending the challenge message.

Lemma 3. *If $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ is an $(16t, t, 1/8)$ approximately incompressible circuit family, then for any PPT simulator $\mathcal{S}$, the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ outputs 1 with probability at most $\mathsf{negl}(\lambda)$.*

Proof. Suppose there exists a simulator $\mathcal{S}$ such that our adversary $\mathcal{A}$ outputs 1 with a non-negligible probability η. We will use $\mathcal{S}$ to show that $\mathcal{C}$ is $(16t, t, 1/8)$ approximately compressible. In particular, we will use $\mathcal{S}$ and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ to construct Cmp and DeCmp circuits satisfying the three properties of an approximately compressible circuit family.

Note that $\mathcal{A}_1$ picks $C_{n_{\mathsf{eval}}+1}, \ldots, C_{n_{\mathsf{eval}}+n_{\mathsf{test}}}$ and x^* uniformly at random and independent of its other choices. Let $r_{\mathcal{S}}$ and $r_{\mathcal{A}}$ denote the randomness used by the simulator $\mathcal{S}$ and adversary $\mathcal{A}$ (in choosing circuits $C_1, \ldots, C_{n_{\mathsf{eval}}}$, and in RO query collection 1 and test phases), respectively. The compression circuit takes as input $(C_1, \ldots, C_{16t}, y_1, \ldots, y_{16t})$, has a randomly chosen string for $r_{\mathcal{S}}$ and $r_{\mathcal{A}}$ hardwired, and works as follows:

- Use $\mathcal{S}$ to generate a public key pk. Give pk to $\mathcal{A}_1$.
- Use $\mathcal{S}$ to generate secrets keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_{n_{\mathsf{key}}}$ for $C'_1, \ldots, C'_{n_{\mathsf{eval}}}, C_1, \ldots, C_{16t}$, where $C'_1, \ldots, C'_{n_{\mathsf{eval}}}$ are sampled using $r_{\mathcal{A}}$. Give the secret keys to $\mathcal{A}_1$.
- Run the first phase of RO query collection. When $\mathcal{A}_1$ makes an RO query in this phase, forward it to $\mathcal{S}$. Give $\mathcal{S}$'s response back to $\mathcal{A}_1$.
- Provide $y_1, \ldots, y_{16t}$ to $\mathcal{S}$. It generates a ciphertext ct^*.
- Run the second phase of RO query collection. Respond to $\mathcal{A}_2$'s RO queries in the same way as before. Let $z_1, \ldots, z_v$ be the responses in order, where $z_i \in \{0,1\}^m$.
- Output ct^* and $z_1, \ldots, z_v$.

The decompression circuit takes $C_1, \ldots, C_{16t}$ and the compressed string $\mathsf{str\text{-}cmp}$ as inputs, which can be parsed as $\mathsf{str\text{-}cmp} = (\mathsf{ct}^*, \{z_i\})$. It also has the random value chosen before for $r_{\mathcal{S}}$ and $r_{\mathcal{A}}$ hardwired, and works as follows:

- Use $\mathcal{S}$ to generate pk and secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_{n_{\mathsf{key}}}$ as before. Give both to $\mathcal{A}_1$.
- Run the first phase of RO query collection. Respond to $\mathcal{A}_1$'s RO queries in the same way as before. Let Γ be the list of RO queries and responses recorded in this phase.
- Run the second phase of RO query collection, where $\mathsf{sk}_1, \ldots, \mathsf{sk}_{n_{\mathsf{eval}}}$ are used to decrypt ct^*. The RO responses required in this step are available as part of the input $(z_1, \ldots, z_v)$. They are also added to Γ.
- Run the test phase with the help of Γ. Let y'_i denote the outcome of decrypting ct^* with $\mathsf{sk}_{n_{\mathsf{eval}}+i}$ for $i \in [n_{\mathsf{test}}]$.
- Output $y'_1, \ldots, y'_{16t}$.

First, note that the size of both compression and decompression circuit is bounded by a polynomial in λ. Next, the output length of the compression circuit is at most $\ell_{\mathsf{ct}} + v \cdot m$, but v is no more than $n_{\mathsf{eval}} \cdot q_{\mathsf{Dec}}$. Thus the output length is bounded by t.

Finally, we need to show that the decompression property works with probability η. When $C_1, \ldots, C_{16t}$ are chosen uniformly at random and $y_1, \ldots, y_{16t}$ is the evaluation of these circuits on a randomly chosen point, then it is easy to see that the decompression circuit emulates the ideal world experiment perfectly. We know that $\mathcal{A}_2$ outputs 1 if and only if for at least 7/8th of the decryptions, $y'_i = y_i$. Hence, if 1 is output with probability η, then the hamming distance of $\mathsf{DeCmp}(\{C_i\}, \mathsf{Cmp}(\{C_i\}, \{y_i\}))$ and $\{y_i\}$ is at most 1/8 with probability at least η.

6 Simulation Secure FE for Bounded Collusions

In this section, we will show an FE scheme that is $(q_1, \mathsf{poly}, \mathsf{q_2})$ simulation secure in the random oracle model, where q_1, q_2 are a-priori fixed polynomials. Since both the pre-challenge and post-challenge queries are bounded, we will simply refer to the total number of key queries. An FE scheme is q-key poly-ciphertext secure if it is $(q_1, \mathsf{poly}, \mathsf{q_2})$ simulation secure as in Definition 8 for all non-negative integers q_1, q_2 s.t. $q_1 + q_2 = q$. We first show a scheme that can handle 1 key query in Section 6.1. Then, in Section 6.2 (and the full version of our paper [1]), we show how to transform a 1-key poly-ciphertext scheme to one that is q-key poly-ciphertext simulation secure for an a-priori fixed q, by first building a scheme for log-depth circuits and then for all poly-size circuits. This transformation is very similar to the one showed by Gorbunov et al. [21], except that they dealt with only one ciphertext.

6.1 Simulation Secure FE for One Key Query

We will now describe our 1-key poly-ciphertext scheme. Recall that in the standard model, it is impossible to have simulation security even for IBE if the adversary is allowed to query for an unbounded number of ciphertexts, followed by one adaptive key query [12, 5]. Here, we show how the random oracle can be used to bypass this impossibility result. At a high-level, the construction is similar to the Sahai-Seyalioglu [31] construction of single-key secure FE from PKE.

Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ be a class of circuits, where each circuit $C \in \mathcal{C}_\lambda$ takes an $n(\lambda)$ bit input and produces an $m(\lambda)$ bit output, and can be represented using $t(\lambda)$ bits. For $x \in \{0,1\}^{n(\lambda)}$, let $U_x^{(\lambda)}$ be a universal circuit that takes any $C \in \mathcal{C}_\lambda$ as input and outputs $C(x)$. Let $\mathcal{U} = \{\mathcal{U}_\lambda\}_\lambda$ be a circuit family such that $\mathcal{U}_\lambda = \{U_x^{(\lambda)} \mid x \in \{0,1\}^{n(\lambda)}\}$. Our one-bounded FE scheme $\mathsf{One\text{-}FE} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{KeyGen}, \mathsf{Decrypt})$ uses a decomposable randomized encoding scheme $(\mathsf{RE.Encode}, \mathsf{RE.Decode})$ for $\mathcal{U}$ and a public key encryption scheme $\mathsf{PKE} = (\mathsf{Setup}_{\mathrm{PKE}},$

$\mathsf{Enc}_{\mathrm{PKE}}, \mathsf{Dec}_{\mathrm{PKE}}$) that can operate on messages of length λ. For simplicity of presentation, we will skip the dependence on λ.

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$: The setup algorithm chooses $2t$ PKE public key/secret key pairs $(\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}) \leftarrow \mathsf{Setup}_{\mathrm{PKE}}(1^\lambda)$ for $i \in [t], b \in \{0,1\}$. It sets $\mathsf{mpk} = \left\{\mathsf{pk}_{i,b}\right\}_{i\in[t],b\in\{0,1\}}$ and $\mathsf{msk} = \{\mathsf{sk}_{i,b}\}_{i\in[t],b\in\{0,1\}}$.
- $\mathsf{Enc}(\mathsf{mpk}, x) \to \mathsf{ct}$: The encryption algorithm first chooses $2t$ random strings $r_{i,b} \leftarrow \{0,1\}^\lambda$ for all $i \in [t]$, $b \in \{0,1\}$. Next, it computes a randomized encoding for the universal circuit U_x, i.e., $\{w_{i,b}\}_{i\in[t],b\in\{0,1\}} \leftarrow \mathsf{RE.Encode}(1^\lambda, U_x)$. Now, let $\mathsf{ct}_{i,b} = \mathsf{Enc}_{\mathrm{PKE}}(\mathsf{pk}_{i,b}, r_{i,b})$ and $\widetilde{\mathsf{ct}}_{i,b} = w_{i,b} \oplus \mathcal{O}(r_{i,b})$ for all $i \in [t]$, $b \in \{0,1\}$. The algorithm outputs $\mathsf{ct} = \left\{\mathsf{ct}_{i,b}, \widetilde{\mathsf{ct}}_{i,b}\right\}_{i\in[t],b\in\{0,1\}}$.
- $\mathsf{KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$: Let $(\beta_1, \ldots, \beta_t)$ be the bit representation of circuit C. The key generation algorithm outputs $\{\mathsf{sk}_{i,\beta_i}\}_{i\in[t]}$ as the secret key for C.
- $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_C, \mathsf{ct})$: Let $\mathsf{sk}_C = \{\mathsf{sk}_{i,\beta_i}\}_{i\in[t]}$ and $\mathsf{ct} = \left\{\mathsf{ct}_{i,b}, \widetilde{\mathsf{ct}}_{i,b}\right\}_{i\in[t],b\in\{0,1\}}$. The decryption algorithm first decrypts the relevant randomized encoding components, i.e., for each $i \in [t]$, it computes $r_{i,\beta_i} = \mathsf{Dec}_{\mathrm{PKE}}(\mathsf{sk}_{i,\beta_i}, \mathsf{ct}_{i,\beta_i})$ and $w_{i,\beta_i} = \widetilde{\mathsf{ct}}_{i,\beta_i} \oplus \mathcal{O}(r_{i,\beta_i})$. Finally, it outputs $\mathsf{RE.Decode}(\{w_{i,\beta_i}\}_{i\in[t]})$.

The correctness of our scheme follows directly from the correctness of the randomized encoding scheme and the public key encryption scheme.

The simulator description and proof of security is given in the full version of our paper [1].

6.2 Simulation Secure FE with Bounded Key Queries for NC1

In this section, we will show how to transform a scheme that handles one key query to one that handles a bounded number of key queries for the class of log-depth circuits. This transformation is identical to the one in [21]. However, the proof is slightly different because we handle unbounded challenge ciphertext queries.

Formal Description Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ be a class of circuits, where each circuit $C \in \mathcal{C}_\lambda$ takes $n(\lambda)$ bit inputs, outputs a single bit and can be represented using an $n(\lambda)$ variate polynomial of degree $D(\lambda)$ over a (large enough) field $\mathbb{F}$. Let q denote a bound on the number of secret key queries. Our FE scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ uses a 1-key poly-ciphertext simulation secure FE scheme ($\mathsf{Setup}_{\mathrm{one}}$, $\mathsf{Encrypt}_{\mathrm{one}}$, $\mathsf{KeyGen}_{\mathrm{one}}$, $\mathsf{Decrypt}_{\mathrm{one}}$) as a building block. Our scheme is parameterized by four polynomials: N, S, v and t, whose values depend on D and q. As in GVW, we set $t(\lambda) = \Theta(q^2\lambda)$, $N(\lambda) = \Theta(N^2q^2t)$ and $v(\lambda) = \Theta(\lambda)$ and $S(\lambda) = \Theta(vq^2)$. We will skip the dependence on λ when it is clear from the context.

For any circuit $C \in \mathcal{C}_\lambda$ and set $\Delta \subset [S]$, we define a circuit $G_{C,\Delta}$ which takes $n + S$ bit inputs and works as follows:

$$G_{C,\Delta}(x_1, \ldots, x_n, y_1, \ldots, y_S) = C(x_1, \ldots, x_n) + \sum_{h\in\Delta} y_h$$

Let $\mathcal{O} = \mathcal{O}_1 \times \ldots \mathcal{O}_N$ be a hash function, where each $\mathcal{O}_i : \{0,1\}^\ell \to \{0,1\}^m$. Each of these hash functions $\mathcal{O}_i$ will be modeled as a random oracle in our security proof.

- $\mathsf{Setup}^{\mathcal{O}}(1^\lambda) \to (\mathsf{MPK}, \mathsf{MSK})$: The setup algorithm runs the one-key FE scheme's setup N times. Let $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{Setup}_{\mathrm{one}}^{\mathcal{O}_i}(1^\lambda)$. The master public key MPK is set to be $\{\mathsf{mpk}_i\}_{i\in[N]}$, and the master secret key MSK is $\{\mathsf{msk}_i\}_{i\in[N]}$.
- $\mathsf{Enc}^{\mathcal{O}}(\mathsf{MPK}, x) \to \mathsf{ct}$: Let $\mathsf{MPK} = \{\mathsf{mpk}_i\}_{i\in[N]}$ and $x = (x_1, \ldots, x_n)$. The encryption algorithm works as follows:
  - It chooses n uniformly random polynomials $\mu_1, \ldots, \mu_n$ of degree t over field $\mathbb{F}$ subject to the constraint that the constant term of μ_i is x_i.
  - It chooses S uniformly random polynomials $\zeta_1, \ldots, \zeta_S$ of degree Dt over field $\mathbb{F}$ and constant term 0.
  - It computes N ciphertexts using the $\mathsf{Encrypt}_{\mathrm{one}}$ algorithm. For $i \in [N]$, it computes $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}_{\mathrm{one}}^{\mathcal{O}_i}(\mathsf{mpk}_i, (\mu_1(i), \ldots, \mu_n(i), \zeta_1(i), \ldots, \zeta_S(i)))$.

  The encryption algorithm outputs $(\mathsf{ct}_1, \ldots, \mathsf{ct}_N)$ as the final ciphertext.
- $\mathsf{KeyGen}^{\mathcal{O}}(\mathsf{MSK}, C)$: Let $\mathsf{MSK} = \{\mathsf{msk}_i\}_{i\in[N]}$. The key generation algorithm works as follows:
  - It chooses a uniformly random set $\Gamma \subset [N]$ of size $Dt + 1$.
  - It chooses a uniformly random set $\Delta \subset [S]$ of size v.
  - It uses the $\mathsf{KeyGen}_{\mathrm{one}}$ algorithm to generate $Dt + 1$ secret keys for the function $G_{C,\Delta}$. For $i \in \Gamma$, it computes $\mathsf{sk}_i \leftarrow \mathsf{KeyGen}_{\mathrm{one}}^{\mathcal{O}_i}(\mathsf{msk}_i, G_{C,\Delta})$.

  The key generation algorithm outputs $(\Gamma, \Delta, \{\mathsf{sk}_i\}_{i\in\Gamma})$ as the secret key for C.
- $\mathsf{Dec}^{\mathcal{O}}(\mathsf{sk}, \mathsf{ct})$: Let $\mathsf{sk} = (\Gamma, \Delta, \{\mathsf{sk}_i\}_{i\in\Gamma})$ and $\mathsf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_N)$. The decryption algorithm works as follows:
  - For each $i \in \Gamma$, let $\alpha_i = \mathsf{Decrypt}_{\mathrm{one}}^{\mathcal{O}_i}(\mathsf{sk}_i, \mathsf{ct}_i)$.
  - It computes a polynomial η of degree Dt over field $\mathbb{F}$ such that for all $i \in \Gamma$, $\eta(i) = \alpha_i$.

  The decryption algorithm outputs $\eta(0^{n+S})$ as the final decryption.

Correctness The correctness proof is identical to the one in [21]. Let $\mu_1, \ldots, \mu_n$, $\zeta_1, \ldots, \zeta_S$ be the polynomials chosen during encryption, and let Γ, Δ be the sets chosen during key generation. From the correctness of the one-key FE scheme, it follows that the decryption algorithm computes $\alpha_i = C(\mu_1(i), \ldots, \mu_n(i)) + \sum_{j\in\Delta} \zeta_j(i)$ for all $i \in \Gamma$. Now, since the polynomial $\eta = C(\mu_1, \ldots, \mu_n) + \sum_{j\in\Gamma} \zeta_j$ has degree Dt and $|\Gamma| = Dt + 1$, the decryption algorithm can compute the polynomial η using the set $\{\alpha_i\}_{i\in[N]}$. Finally, note that $\eta(0^{n+S}) = C(\mu_1(0), \ldots, \mu_n(0)) + \sum_j \zeta_j(0) = C(x_1, \ldots, x_n)$.

In the full version of our paper [1], we prove security of our scheme for NC1 and describe how this scheme can be bootstrapped to all poly-size circuits.

References

1. Agrawal, S., Koppula, V., Waters, B.: Impossibility of simulation secure functional encryption even with random oracles. Cryptology ePrint Archive, Report 2016/959 (2016), https://eprint.iacr.org/2016/959
2. Agrawal, S., Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption: New perspectives and lower bounds. In: CRYPTO (2013)
3. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: CRYPTO (2015)
4. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. Computational Complexity 15(2), 115–162 (2006)
5. Bellare, M., O'Neill, A.: Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In: CANS (2013)
6. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security. pp. 62–73 (1993)
7. Bitansky, N., Lin, H., Paneth, O.: On removing graded encodings from functional encryption. In: Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II. pp. 3–29 (2017)
8. Bitansky, N., Nishimaki, R., Passelègue, A., Wichs, D.: From cryptomania to obfustopia through secret-key functional encryption. In: Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II. pp. 391–418 (2016)
9. Bitansky, N., Paneth, O.: On the impossibility of approximate obfuscation and applications to resettable cryptography. In: STOC (2013)
10. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: FOCS (2015)
11. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil Pairing. In: CRYPTO (2001)
12. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: TCC (2011)
13. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: TCC (2007)
14. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. of the ACM 51(4), 557–594 (2004)
15. Canetti, R., Kalai, Y.T., Paneth, O.: On obfuscation with random oracles. In: TCC (2015)
16. Caro, A.D., Jain, V.I.A., O'Neill, A., Paneth, O., Persiano, G.: On the achievability of simulation-based security for functional encryption. In: CRYPTO (2013)
17. Caro, A.D., Jain, V.I.A., O'Neill, A., Paneth, O., Persiano, G.: On the achievability of simulation-based security for functional encryption. Cryptology ePrint Archive, Report 2013/364 (2013)
18. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: FOCS. pp. 464–479 (1984)
19. Goldwasser, S., Kalai, Y., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Succinct functional encryption and applications: Reusable garbled circuits and beyond. In: STOC (2013)

20. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: CRYPTO (2012)
21. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: STOC (2013)
22. Hubácek, P., Wichs, D.: On the communication complexity of secure function evaluation with long output. In: Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015. pp. 163–172 (2015), http://doi.acm.org/10.1145/2688073.2688105
23. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA. pp. 44–61 (1989)
24. Iovino, V., Zebroski, K.: Simulation-based secure functional encryption in the random oracle model. In: LATINCRYPT (2015)
25. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: EUROCRYPT (2008)
26. Lin, H., Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation with non-trivial efficiency. In: Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part II. pp. 447–462 (2016)
27. Mahmoody, M., Mohammed, A., Nematihaji, S., Pass, R., Shelat, A.: Lower bounds on assumptions behind indistinguishability obfuscation. In: TCC (2016)
28. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: CRYPTO (2002)
29. O'Neill, A.: Definitional issues in functional encryption. IACR Cryptology ePrint Archive 2010, 556 (2010), http://eprint.iacr.org/2010/556
30. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010)
31. Sahai, A., Seyalioglu, H.: Worry-free encryption: functional encryption with public keys. In: ACM CCS (2010)
32. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: EUROCRYPT. pp. 457–473 (2005)
33. Shamir, A.: Identity-based cryptosystems and signature schemes. In: CRYPTO (1984)
34. Yao, A.: How to generate and exchange secrets. In: FOCS. pp. 162–167 (1986)