Upgrading to Functional Encryption

Saikrishna Badrinarayanan[1⋆], Dakshita Khurana[2⋆⋆], Amit Sahai[1⋆⋆⋆], and Brent Waters[3†]

[1] UCLA
{saikrishna, sahai}@cs.ucla.edu
[2] MSR New England
dakshkhurana@gmail.com
[3] UT Austin
bwaters@cs.utexas.edu

Abstract. The notion of Functional Encryption (FE) has recently emerged as a strong primitive with several exciting applications. In this work, we initiate the study of the following question: Can existing public key encryption schemes be "upgraded" to Functional Encryption schemes without changing their public keys or the encryption algorithm? We call a public-key encryption scheme with this property to be *FE-compatible*. Indeed, assuming ideal obfuscation, it is easy to see that every CCA-secure public-key encryption scheme is FE-compatible. Despite the recent success in using indistinguishability obfuscation to replace ideal obfuscation for many applications, we show that this phenomenon most likely will not apply here. We show that assuming fully homomorphic encryption and the learning with errors (LWE) assumption, there exists a CCA-secure encryption scheme that is provably *not FE-compatible*. We also show that a large class of natural CCA-secure encryption schemes proven secure in the random oracle model are not FE-compatible in the random oracle model.

⋆ Research supported in part by the IBM PhD Fellowship.

⋆⋆ Research done while at UCLA, supported in part by the UCLA Dissertation Year Fellowship.

⋆⋆⋆ Research of first, second and third author supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

† Research supported by NSF CNS-1228599 and CNS-1414082, DARPA SafeWare, Microsoft Faculty Fellowship, and Packard Foundation Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense or the U.S. Government.

Nevertheless, we identify a key structure that, if present, is sufficient to provide FE-compatibility. Specifically, we show that assuming sub-exponentially secure iO and sub-exponentially secure one way functions, there exists a class of public key encryption schemes which we call *Special-CCA* secure encryption schemes that are in fact, FE-compatible. In particular, each of the following popular CCA secure encryption schemes (some of which existed even before the notion of FE was introduced) fall into the class of *Special-CCA* secure encryption schemes and are thus FE-compatible:

1. [CHK04] when instantiated with the IBE scheme of [BB04].
2. [CHK04] when instantiated with any Hierarchical IBE scheme.
3. [PW08] when instantiated with any Lossy Trapdoor Function.

1 Introduction

Functional Encryption (FE) [SW05,SW08] is a powerful framework that significantly expands the scope of public-key encryption. In an ordinary public-key encryption scheme, a user Alice first chooses a public key PK and a corresponding secret key SK using a (master) setup algorithm Setup. Then, any other user Bob can use Alice's public key to encrypt a message m to obtain a ciphertext $c = \mathsf{Enc}(\mathsf{PK}, m)$. Alice can decrypt this ciphertext using her secret key, yielding $m = \mathsf{Dec}(\mathsf{SK}, c)$.

In a functional encryption scheme, we give Alice key delegation capabilities: Alice can use a new key generation algorithm KeyGen to generate a *functional key* $\mathsf{SK}_f = \mathsf{FE.KeyGen}(\mathsf{SK}, f)$ for a function f that is, say, described by a circuit. Then Alice can hand this functional key SK_f to an associate Charlie, and Charlie can use this functional key together with a new decryption algorithm to only learn $f(m) = \mathsf{FE.Dec}(\mathsf{SK}_f, c)$ when given the ciphertext c. Intuitively speaking, nothing[4] beyond $f(m)$ should be learned by Charlie when given SK_f and c. This notion was fully formalized by [BSW11] in the setting where many functional keys and ciphertexts may be given to an adversary. The first work achieving functional encryption for general functions was [GGH$^+$13], using the power of indistinguishability obfuscation.

The work of [BSW11] gave several compelling applications of functional encryption. For instance, Alice may want to store her e-mail in encrypted form, but she wants her cloud provider to be able to execute a phishing-detection circuit C on her email prior to sending it to her for decryption. She could accomplish this goal by providing her cloud provider with a functional key for SK_C, and the only thing the cloud provider would learn is whether any email received by Alice satisfies the phishing-detection circuit.

Applications of functional encryption become even more compelling when we think of Alice as representing a large organization or company. In such a

[4] Slightly more formally, functional encryption requires that encryptions of two messages m_0 and m_1 should be indistinguishable when given functional keys corresponding to any functions f that satisfy $f(m_0) = f(m_1)$. See Section 3 for more details.

scenario, the threat that functional encryption helps to address cryptographically is the insider threat. For example, consider an organization like a government tax authority, that regularly handles extremely sensitive information, but where individuals within the organization should only have access to limited digests or snippets of this sensitive information. For example, an analyst Dave at the tax authority may need only to compute statistical summaries of tax returns filed by a large set of people. Functional encryption would allow Dave to obtain a functional key SK_T, where T is the description of a function that produces statistical summaries of tax returns. The security of functional encryption would guarantee that even if Dave goes rogue, Dave's functional key would only allow him to learn and exfiltrate statistical summaries, and not any more personal information about individual tax returns beyond what could be deduced from the statistical summary.

Contrast this to the case where only ordinary public-key encryption is used to encrypt tax information. In this case, Dave would need the (master) secret key SK in order to decrypt tax information before processing it to obtain statistical summaries. And therefore a rogue Dave could exfiltrate the personal details of any person's tax return that was an input to the statistical summary he was supposed to compute. This is just one example, illustrative of many such scenarios where functional encryption could be beneficial for security.

Upgrading to Functional Encryption. Suppose that some time in the future, an organization, upon hearing about the advantages of functional encryption, wishes to "upgrade" to use functional encryption. Such an organization may face many challenges. First, the organization may already have infrastructure in place where partners and clients use an existing public-key encryption scheme to communicate with the organization. As such, the organization may have already amassed large amounts of encrypted data using a legacy public-key encryption system. Second, the organization may face regulatory burdens like HIPAA or other future regulations, that require the organization to use a particular encryption algorithm. Third, it could be that, even in this future time, existing key generation algorithms for general-purpose functional encryption (which typically currently use indistinguishability obfuscation) are too slow, but the organization wants to be ready for the day when such algorithms become practical.

In light of these concerns, what public-key encryption algorithm should the organization use now? While these are mostly societal challenges, security must exist in the context of human societies with traditions, rules, and regulations. And in this case, these concerns give rise to an intriguing theoretical question:

What (existing) public-key encryption algorithms can be "upgraded" to become functional encryption schemes, without changing the encryption algorithm or the public keys?

Our paper initiates the systematic study of this question. To formalize this, we say that a public-key encryption scheme E is *FE-compatible* if there exist new key generation and decryption algorithms that, when combined with the original setup and encryption algorithms of E, yield a (selectively) secure functional encryption scheme. (See Section 3 for details.)

Necessary Conditions. The technical starting point for our work is the folklore observation that any functional encryption scheme must satisfy a certain level of non-malleability. To see why, consider a functional encryption scheme for encrypting $(n+1)$-bit messages m, and consider the function f_1 that on input m simply outputs the first n bits of m. Suppose that we obtain a functional key SK_{f_1} for this function. Then functional encryption guarantees that encryptions of any two messages with identical n-bit prefixes should still be indistinguishable from each other.

But suppose there was a way for an adversary to modify any encryption $\mathsf{FE.Enc}(m)$ to obtain $\mathsf{FE.Enc}(m')$ where m' swapped the first and last bits of m. This would, for example, easily be possible if one tried to encrypt the message bit-by-bit. Then, by applying the functional key SK_{f_1} to $\mathsf{FE.Enc}(m')$, the adversary would learn the last bit of m, and break the security that is supposed to be guaranteed by functional encryption.

Indeed, it is not hard to see that the above argument generalizes to guarantee a type of security against chosen-ciphertext attacks. Thus, (a form of) CCA-security is a necessary requirement for an encryption scheme to be FE-compatible.

Universal Functional Encryption? At this point, it might be tempting to consider the possibility that CCA-security is also a *sufficient* condition for being FE-compatible. Indeed, this would be true if we had ideal obfuscation[5] [Had00] – that is, obfuscation that creates the equivalent of a virtual black box. It is not difficult to see why: To create a functional key SK_f, simply obfuscate the function that uses SK as a hardwired constant to decrypt the input ciphertext c to obtain the message m, and then simply output $f(m)$. If the obfuscation is ideal, then this functional key can easily be simulated as a black box just by using the CCA-decryption oracle for decryption. Thus, given ideal obfuscation, every CCA-secure public-key encryption scheme is FE-compatible. In this sense, we could hope to have a kind of universal functional encryption (in the sense of universal deniable encryption [SW14] or universal signature aggregators [HKW15]), where the key generation construction above could be applied to any CCA-secure encryption scheme.

Recently our field has had remarkable success in achieving results using indistinguishability obfuscation that were previously known to be possible only using ideal obfuscation, especially using the punctured programming paradigm of [SW14]. Is this just a matter of applying enough "$i\mathcal{O}$ gymnastics" to make this work?

Our Results. In our first result, somewhat surprisingly, we show that in this case, the intuition based on ideal obfuscation is wrong. Specifically, we show the following:

[5] Note that ideal obfuscation is impossible to build.

Informal Theorem 1 *Assuming CCA-secure public-key encryption, fully homomorphic encryption (FHE) and LWE, there exists a CCA-secure public-key encryption scheme that is provably not FE-compatible.*

The construction we give in the impossibility result above is quite contrived, like most impossibility results of this type. Could it be that all "natural" CCA-secure public-key encryption schemes are FE-compatible? Sadly, we do not know how to answer, or even formally define, this question. Nevertheless, one natural setting in which to consider this question is the well-studied random oracle model; this model allows for very simple and intuitive proofs of CCA-security, via the popular Fujisaki-Okamoto [FO99] transformation. In the random oracle model, however, we show an even stronger negative result: Every public-key encryption scheme, when converted into a CCA-secure encryption scheme in the random oracle model via the Fujisaki-Okamoto transformation, is provably not FE-compatible in the random oracle model. Thus, in the random oracle model, we obtain a large family of natural CCA-secure schemes [6] that are not FE-compatible. [7]

In light of the impossibility results above, we believe that a systematic study of FE-compatibility will need to proceed in a "bottom-up" manner, by looking at existing classes of CCA-secure encryption schemes and seeing if they can indeed be FE-compatible. We initiate this line of study by identifying a key structure that, if present, is sufficient to provide FE-compatibility. Specifically, we show the following:

Informal Theorem 2 *Assuming sub-exponentially secure one way functions and sub-exponentially secure iO, there exists a class of public key encryption schemes which we call* Special-CCA *secure encryption schemes that are FE-compatible.*

We then note that several existing CCA-secure encryption schemes fall into the class of *Special-CCA* secure encryption schemes. As a result, we get the following theorem:

Informal Theorem 3 *Assuming sub-exponentially secure indistinguishability obfuscation and sub-exponentially secure one way functions, each of the following existing CCA-secure encryption schemes are FE-compatible:*

- [*CHK04*] *when instantiated with the IBE scheme of* [*BB04*].
- [*CHK04*] *when instantiated with any Hierarchical IBE scheme.*
- [*PW08*] *when instantiated with any Lossy Trapdoor Function.*

[6] We believe similarly structured transformation such as RSA-OAEP [BR94] will have the same issues.

[7] Interestingly, if the scheme is instantiated with a particular hash function family it might actually be FE-compatible. This is somewhat the opposite of a typical RO infeasibility results where one usually finds a scheme is provably secure in the RO model, but is insecure under any concrete instantiation. Unfortunately, it is unclear how to argue positive security of any such concrete FO instantiations as the usual RO heuristic is now off limits.

It is interesting to note that the above CCA-secure encryption schemes are each at least 9 years old, and yet they can be used to build functional encryption schemes without changing the encryption mechanism. Contrast this to existing functional encryption schemes before our work, most of which have specifically designed encryption methods using "$i\mathcal{O}$-friendly" tools.

Finally, we also consider a weaker notion called *key-only FE-compatibility* where we retain only the public key and secret key of the public key encryption scheme and design new encryption, function secret key generation and decryption algorithms to "upgrade" it to a FE scheme. In the common random string model, we show that assuming polynomially hard iO, every public key encryption scheme is key-only FE compatible - that is, it can be upgraded to a selectively secure FE scheme for any function family. We refer the reader to the full version for details regarding this notion and the corresponding results we achieve.

Open problems and future work. It would be interesting to understand if there exists other classes of encryption schemes that are FE-compatible. More generally, an interesting open problem would be to study what is the exact type of CCA-security needed for an encryption scheme to be FE-compatible.

While it is known that general purpose functional encryption implies indistinguishability obfuscation, another interesting direction would be to weaken the security requirement of functional encryption (for example, bounded-key secure FE) and understand what class of encryption schemes can be upgraded without the use of indistinguishability obfuscation. A solution in this setting might also be practical in today's world. Going in the other direction, an interesting feasibility question is whether we can upgrade existing encryption schemes to achieve general purpose multi-input functional encryption [GGG+14,BGJS15].

Finally, we observe that in our positive result, on upgrading the CCA secure encryption schemes into an FE scheme, it may potentially lose the CCA property. It is an interesting open problem to define and achieve FE-CCA compatibility[8].

2 Technical Overview

The question at the core of this paper is: what kinds of public-key encryption schemes can be "upgraded" to yield functional encryption schemes? Informally speaking, we say that a public-key encryption scheme PKE is *FE-compatible* if a functional encryption scheme can be generated where the setup and encryption algorithms of the functional encryption scheme are the same as the public-key encryption scheme. Namely, we have $\mathsf{FE.Setup} = \mathsf{PKE.Setup}$ and $\mathsf{FE.Enc} = \mathsf{PKE.Enc}$. Thus, only the functional encryption key generation and decryption algorithms are allowed to be newly specified.

As already noted, the technical starting point for our work is the folklore observation that any functional encryption scheme must satisfy a certain level of non-malleability. To remind ourselves why, consider a functional encryption

[8] Note that our negative result would still hold in this stronger model of FE-CCA compatibility.

scheme for encrypting $(n+1)$-bit messages m, and consider the function f_1 that on input m simply outputs the first n bits of m. Suppose that we obtain a functional key SK_{f_1} for this function. Then functional encryption guarantees that encryptions of any two messages with identical n-bit prefixes should still be indistinguishable from each other.

But suppose there was a way for an adversary to modify any encryption $\mathsf{FE.Enc}(m)$ to obtain $\mathsf{FE.Enc}(m')$ where m' swapped the first and last bits of m. This would, for example, easily be possible if one tried to encrypt the message bit-by-bit. Then, by applying the functional key SK_{f_1} to $\mathsf{FE.Enc}(m')$, the adversary would learn the last bit of m, and break the security that is supposed to be guaranteed by functional encryption.

An impossibility result. The most natural question to ask, then, is whether CCA-security is also a sufficient condition for FE-compatibility. In our first result, we prove that this is indeed not the case: we construct a counterexample public-key encryption scheme that satisfies CCA-security, but provably is not FE-compatible.

Let us build some intuition for how our impossibility result will proceed. The main difference between the CCA security game and the FE security game is that in the CCA security game, there is a decryption *oracle*, whereas in the FE security game, the adversary can actually obtain a circuit that will (at least partially) decrypt ciphertexts. This is reminiscent of the situation underlying the impossibility result of Barak et al. [BGI+01] for virtual black-box obfuscation: There, the ideal model gave oracle access to the function to be obfuscated, whereas the real model gave the adversary an actual circuit implementing that function. Indeed, we draw inspiration from [BGI+01] in devising our negative result, although we differ from it in almost every technical respect.

The idea behind our negative result will be to take an arbitrary CCA-secure encryption scheme $(\mathsf{Setup}_{\mathsf{CCA}}, \mathsf{Enc}_{\mathsf{CCA}}, \mathsf{Dec}_{\mathsf{CCA}})$ and somehow "damage" it to make it FE-incompatible, without disturbing its CCA security. This "damaged scheme" must somehow make use of the fact that an FE-adversary will be able to ask for and obtain a functional key SK_{f_1}, let us say for the same prefix-revealing function f_1 that we defined above. This functional key SK_{f_1} enables the FE-adversary to compute a prefix-decryption circuit D that outputs the first n bits of the message corresponding to any ciphertext.

Our first idea (which conceptually dates back to [BGI+01]) is to use fully homomorphic encryption (FHE) to help us take advantage of this situation. We first choose a random n-bit string α, and encrypt it $c = \mathsf{Enc}_{\mathsf{CCA}}(\alpha||0)$ using the CCA-secure encryption scheme. But then we re-encrypt this $c' = \mathsf{FHE}(c)$ using the fully homomorphic encryption scheme. We reveal c' as part of the public key of the "damaged scheme," but crucially both α and c are kept hidden.

Why does this help? Because now an FE-adversary that obtains the prefix-decryption circuit D can compute $\mathsf{FHE.Eval}(D, c') = \mathsf{FHE}(\alpha)$. While it is not yet clear that this is useful for any attack, we observe that, at least intuitively, a CCA-attacker has no obvious way to obtaining $\mathsf{FHE}(\alpha)$ from the public key and the decryption oracle (though formally proving this will be the main technical challenge of our impossibility result, as we will discuss shortly). This is because the only information that the CCA-attacker has about α is contained in c', but c' is an encryption under FHE and the decryption oracle only decrypts ciphertexts validly encrypted using $\mathsf{Enc}_{\mathsf{CCA}}$.

To enable a real attack, then, we also add to the public key an obfuscation of a program P that takes as input an FHE ciphertext e, decrypts it, and checks whether this decryption is equal to α. If so, it outputs the secret key needed for executing $\mathsf{Dec}_{\mathsf{CCA}}$, otherwise it outputs $\perp$. Because the FE-attacker can obtain $\mathsf{FHE}(\alpha)$ as noted above, it can then use the obfuscated program to obtain the full secret key for executing $\mathsf{Dec}_{\mathsf{CCA}}$, breaking the security of the FE scheme.

Why these changes preserve CCA security. The changes above – adding the FHE ciphertext c' and the obfuscated program P to the public key – only provide an impossibility result if CCA security is preserved even after these two objects are added to the public key. While it is not obvious how a CCA-attacker could use these objects to break security, in order to prove CCA security, intuitively we will need to remove the dependence of c' on α. But $c' = \mathsf{FHE}(\mathsf{Enc}_{\mathsf{CCA}}(\alpha||0))$, and the obfuscated program P contains the secret key for FHE. But in order to remove these secret keys from P, intuitively we need to remove the "trigger" point $\mathsf{FHE}(\alpha)$ from the code of P, for which we first need to remove the dependence of c' on α. This chicken-and-egg situation is the primary technical obstacle that we need to overcome to finish the proof.

To deal with this problem, we draw inspiration from the work of Myers and Shelat [MS09] and Hohenberger, Lewko and Waters [HLW12] that considered the seemingly very different problem of converting any CCA-secure encryption scheme for single-bit messages into a CCA-secure encryption scheme for multi-bit messages. However, to implement our inspiration, we will need to make a technical change to the encryption system. Instead of using $\mathsf{Enc}_{\mathsf{CCA}}$ to encrypt the entire $n+1$-bit message, we will use the CCA-secure encryption schemes to encrypt the first n bits of the message, and use a separate encryption scheme $\mathsf{Enc}_{\mathsf{CPA}}$ to encrypt the last bit of the message. (In fact, we will use $\mathsf{Enc}_{\mathsf{CCA}}$ to jointly encrypt the first n bits of the message and the ciphertext produced by $\mathsf{Enc}_{\mathsf{CPA}}$. But we will ignore this detail for the purpose of this overview.) Finally, we will change our obfuscated program P to output just the secret key for executing $\mathsf{Dec}_{\mathsf{CPA}}$ to decrypt the last bit. This way, the secret key for executing $\mathsf{Dec}_{\mathsf{CCA}}$ is independent of the program P. Now, we will define a Bad Event to be when a CCA-attacker queries its decryption oracle on the ciphertext $c = \mathsf{Enc}_{\mathsf{CCA}}(\alpha)$. Looking ahead, we will first consider the situation when this Bad Event does not happen. Then, we will show that indeed the Bad Event can only occur with negligible probability.

Suppose that we know that the Bad Event cannot happen. Then, the decryption oracle given to the CCA-attacker is equivalent to a decryption oracle that would be given to a CCA-attacker if $c = \mathsf{Enc}_{\mathsf{CCA}}(\alpha)$ was the "challenge" ciphertext on which the attacker is not allowed to query. Note that in this case, the CCA security of $\mathsf{Enc}_{\mathsf{CCA}}$ already guarantees that $c = \mathsf{Enc}_{\mathsf{CCA}}(\alpha)$ is indistinguishable from $c = \mathsf{Enc}_{\mathsf{CCA}}(0^n)$, even to an adversary that is given the obfuscated program P as auxiliary information about α. Thus, we can already remove the dependence of c' on α.

Now, the only part of the public key that depends on α is the obfuscated program P, and we just need to get rid of it. This could be accomplished via $i\mathcal{O}$ using the fact that α is a uniformly random string, but in fact our job is made even easier due to the recent works on “lockable obfuscation” of Goyal et al. [GKW17] and Wichs and Zirdelis [WZ17]. These works consider obfuscating programs $C(x)$ whose structure is exactly such that, for some circuit Test if $\mathsf{Test}(x) = \alpha$, then some secret β is revealed, and otherwise the output is $\perp$. Lockable obfuscation states that if α is chosen uniformly (and, for our setting, no auxiliary information about α is revealed), then such obfuscated programs are indistinguishable from obfuscated programs that always output $\perp$ and have no secrets within them whatsoever. Furthermore, such lockable obfuscation is possible to construct just assuming LWE for suitable parameters. Thus, applying the security of lockable obfuscation, we are able to replace the obfuscated program P with a program that always outputs $\perp$, thereby completely removing any information about the secret keys of any encryption scheme and about α. This shows that the new scheme is CCA-secure, under the assumption that the Bad Event does not occur.

All that remains to be done is to prove that the Bad Event does not occur. Counterintuitively, we first observe that the lockable obfuscation argument above already shows that the Bad Event cannot occur if the ciphertext c had been $c = \mathsf{Enc}_{\mathsf{CCA}}(0^n)$ instead of $c = \mathsf{Enc}_{\mathsf{CCA}}(\alpha)$. In other words, if $c = \mathsf{Enc}_{\mathsf{CCA}}(0^n)$, then the adversary never queries the decryption oracle with c. Now, suppose for sake of contradiction, that the adversary does query c with noticeable probability if $c = \mathsf{Enc}_{\mathsf{CCA}}(\alpha)$. Then, we can use this to break CCA-security of $\mathsf{Enc}_{\mathsf{CCA}}$; take as a challenge ciphertext c that is either $c = \mathsf{Enc}_{\mathsf{CCA}}(\alpha)$ or $c = \mathsf{Enc}_{\mathsf{CCA}}(0^n)$. Then run the adversary until it attempts to query the oracle on c. If it ever does this, we can conclude that $c = \mathsf{Enc}_{\mathsf{CCA}}(\alpha)$. If it doesn't, then we can output a random guess. This will give us an nontrivial advantage in determining whether $c = \mathsf{Enc}_{\mathsf{CCA}}(\alpha)$ or $c = \mathsf{Enc}_{\mathsf{CCA}}(0^n)$.

This completes the impossibility proof. Full details can be found in Section 4. For the impossibility result that applies to CCA secure encryption schemes built using the Fujisaki-Okamoto transformation in the random oracle model, we refer the reader to the full version of the paper.

Positive Results for FE-compatibility. Our impossibility result shows that CCA security is not a sufficient condition for an encryption scheme to be FE-compatible. On the other hand, unfortunately positive results on FE in the literature (e.g. [GGH$^+$13,Wat14]) typically construct special-purpose encryption methods that are atypical for achieving CCA security. For instance, even though the original general-purpose FE scheme of [GGH$^+$13] follows the Naor-Yung paradigm [NY90,Sah99], instead of using a simulation-sound NIZK in the encryption, it uses a special object introduced in [GGH$^+$13] called a statistically simulation-sound NIZK. Recall that our goal is to find existing CCA-secure encryption schemes that are already FE-compatible, rather than design special-purpose (sometimes called "$i\mathcal{O}$ friendly") primitives that would enable FE.

How can we go about this? Let us try to see if there are encryption mechanisms that were useful in achieving CCA-security that can also be sufficient for achieving FE.

Our key observation is that the notion of a *punctured* decryption key, which has implicitly been used for building CCA-security for over a decade, since (at least) the work of [CHK04], can also be useful for building FE functional keys. Roughly speaking, we consider the notion of a tag-based encryption, where every ciphertext is associated with a tag. Then, a punctured decryption key $\mathsf{SK}_{\mathsf{tag}^*}$ should allow a user to decrypt every ciphertext with $\mathsf{tag} \neq \mathsf{tag}^*$, but messages encrypted under tag tag^* should still be semantically secure. Intuitively, such punctured keys have been useful for building CCA-secure encryption because a punctured decryption key would allow the implementation of a decryption oracle that would still not be able to decrypt a challenge message that was encrypted under tag tag^*. In the literature, such schemes are combined with one-time signature schemes, where the tag is set to be the verification key of such a one-time signature scheme, and then the ciphertext is signed in a way that verifies with this key.

How can we use this idea for building FE functional keys? At a high level, we start with the most basic idea for building a functional key for a function f. We can simply obfuscate a program that has the decryption key built in, uses this decryption key to decrypt the message m, and then outputs $f(m)$. Now, we need to argue that the encryption of m_0 and the encryption of m_1 should be indistinguishable as long as $f(m_0) = f(m_1) = y$. The first idea is to fix the verification key VK^* in advance that will be used as the tag for the challenge ciphertext c^*. Now, we can reformulate the obfuscated program to first check whether the input ciphertext is equal to c^*, in which case the program should output y, but otherwise it should just use the decryption key to decrypt the message m, and then output $f(m)$ as before. This program is functionally equivalent to the previous one, and therefore indistinguishability of obfuscated programs follows from $i\mathcal{O}$.

Now, our goal will be to replace the decryption key within the program with the punctured decryption key $\mathsf{SK}_{\mathsf{VK}^*}$. However, note that we cannot do that immediately, because there are many valid ciphertexts for various messages m that could be signed under verification key VK^*, on which the program is supposed to output $f(m)$. However, we know that it should be hard for the adversary to actually find such valid ciphertexts, because of the security of the one-time signature scheme. Here, we can use sub-exponentially secure $i\mathcal{O}$ to complete the argument: Roughly speaking, the work of [BCP14] shows that if an $i\mathcal{O}$ scheme is secure against time $T \cdot \mathsf{poly}(\mathsf{n})$ adversaries, then $i\mathcal{O}(P_1)$ and $i\mathcal{O}(P_2)$ are indistinguishable as long as: (1) they only differ on at most T inputs, and (2) these inputs are hard to find even if given the code of both P_1 and P_2, even for machines whose running time far exceed T. By using this, assuming also sub-exponentially secure one-time signatures (which follow from sub-exponentially strong one-way functions), we can replace the program with one that first checks whether the input ciphertext is equal to c^*, in which case the program outputs y, but otherwise it uses the *punctured* decryption key $\mathsf{SK}_{\mathsf{VK}^*}$ to decrypt the message m, and then output $f(m)$ as before.

Now, since only this punctured decryption key $\mathsf{SK}_{\mathsf{VK}^*}$ is used, we can argue that an encryption of m_0 under tag VK^* is indistinguishable from an encryption of m_1 under tag VK^*. Thus, we show how to bootstrap punctured decryption keys as an existing method for building CCA-secure encryption, into a method for constructing functional keys without needing to change the underlying encryption scheme. Interestingly, the security of the encryption given a punctured decryption key needs to hold only against polynomial-time adversaries, as in standard proofs of CCA-security.

We observe that at least three different existing CCA-secure schemes from the literature, some dating back over a decade, already follow the punctured key approach to building CCA-secure encryption, and therefore are FE-compatible. Full details can be found in Section 5.

2.1 Preliminaries and Organization.

We refer the reader to the full version for definitions of the following primitives: public key encryption, indistinguishability obfuscation, differing inputs obfuscation, lockable obfuscation and fully homomorphic encryption.

In Section 3, we define the notion of FE-compatibility. In Section 4 we show the impossibility result. Finally, in Section 5, we show the constructions of FE-compatible CCA secure encryption schemes.

3 Defining Functional Encryption Compatibility

Throughout, let the security parameter be denoted by n. Let $\mathcal{X} = \{\mathcal{X}_n\}_{n\in\mathbf{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_n\}_{n\in\mathbf{N}}$ denote ensembles where each $\mathcal{X}_n$ and $\mathcal{Y}_n$ is a finite set. Let $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbf{N}}$ denote an ensemble where each $\mathcal{F}_n$ is a finite collection of functions, and each function $f \in \mathcal{F}_n$ takes as input a string $x \in \mathcal{X}_n$ and outputs $f(x) \in \mathcal{Y}_n$.

We first define the notion of functional encryption(FE) in the next subsection and then, we define what it means for a public key encryption scheme to be FE-Compatible.

3.1 Functional Encryption

A functional encryption scheme $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.Keygen}, \mathsf{FE.Dec})$ for a family of message spaces $\{\mathcal{X}_n\}$, a family of output spaces $\{\mathcal{Y}_n\}$ and a family of functions $\mathcal{F}$ consists of the following polynomial time algorithms:

– $\mathsf{FE.Setup}(1^n)$. The setup algorithm takes as input the security parameter n and outputs a master public key-secret key pair $(\mathsf{MPK}, \mathsf{MSK})$.

- $\mathsf{FE.Enc}(\mathsf{MPK}, x) \rightarrow \mathsf{CT}$. The encryption algorithm takes as input a message $x \in \mathcal{X}_n$ and the master public key MPK. It outputs a ciphertext CT.
- $\mathsf{FE.Keygen}(\mathsf{MSK}, f) \rightarrow \mathsf{SK}_f$. The key generation algorithm takes as input a function $f \in \mathcal{F}_n$ and the master secret key MSK. It outputs a function secret key SK_f.
- $\mathsf{FE.Dec}(\mathsf{SK}_f, \mathsf{CT}) \rightarrow y$. The decryption algorithm takes as input a secret key SK_f and a ciphertext CT. It outputs a string $y \in \mathcal{Y}_n$ or $\perp$.

Definition 1. *(Correctness) A functional encryption scheme* FE *for* $\mathcal{F}$ *is correct if for all* $f \in \mathcal{F}_n$ *and all* $x \in \mathcal{X}_n$

$$\Pr\begin{bmatrix} (\mathsf{MPK},\mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^{\mathsf{n}}) \\ \mathsf{SK}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{MSK}, f) \\ \mathsf{FE.Dec}(\mathsf{SK}_f, \mathsf{FE.Enc}(\mathsf{MPK}, x)) = f(x) \end{bmatrix} = 1$$

where the probability is over the random coins of $\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.Keygen}$ *and* $\mathsf{FE.Dec}$.

Security We define the security notion for a functional encryption scheme using the following game ($\mathsf{Adaptive} - \mathsf{IND}$) between a challenger and an adversary.

Setup Phase: The challenger generates $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^{\mathsf{n}})$ and then hands over the master public key MPK to the adversary.
Key Query Phase 1: The adversary makes function secret key queries by submitting functions $\mathsf{f} \in \mathcal{F}_n$. The challenger responds by giving the adversary the corresponding function secret key $\mathsf{SK_f} \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}, \mathsf{f})$.
Challenge Phase: The adversary chooses two messages $(\mathsf{m}_0, \mathsf{m}_1)$ of the same size (each in $\mathcal{X}_n$) such that for all queried functions f in the key query phase, it holds that $\mathsf{f}(\mathsf{m}_0) = \mathsf{f}(\mathsf{m}_1)$. The challenger selects a random bit $\mathsf{b} \in \{0, 1\}$ and sends a ciphertext $\mathsf{CT} \leftarrow \mathsf{FE.Enc}(\mathsf{MPK}, \mathsf{m_b})$ to the adversary.
Key Query Phase 2: The adversary may submit additional key queries $\mathsf{f} \in \mathcal{F}_n$ as long as they do not violate the constraint described above. That is, for all queries f, it must hold that $\mathsf{f}(\mathsf{m}_0) = \mathsf{f}(\mathsf{m}_1)$.
Guess: The adversary submits a guess $b^{'}$ and wins if $b^{'} = b$. The adversary's advantage in this game is defined to be $2 * |\Pr[\mathsf{b} = \mathsf{b}^{'}] - 1/2|$.

We also define the *selective* security game, which we call ($\mathsf{Selective} - \mathsf{IND}$) where the adversary outputs the challenge message pair even before seeing the master public key.

Definition 2. *A functional encryption scheme* FE *is selective/adaptive secure if all PPT adversaries have at most a negligible advantage in the* $\mathsf{Selective} - \mathsf{IND}/\mathsf{Adaptive} - \mathsf{IND}$ *security game.*

We can also parameterize by the number of function secret key queries the adversary can make in the security game.
Compactness[AJ15] : A functional encryption scheme is said to be compact if the size of the ciphertext does not depend on the size of the functions

that the scheme can handle. That is, let $p(\cdot)$ be a polynomial. Now, any functional encryption scheme FE for a class of functions $\mathcal{F}$ is said to be compact if $|\mathsf{FE.Enc}(\mathsf{MPK}, x)| = p(n, |x|)$ where n is the security parameter.

3.2 FE-Compatibility

In this section, we define a property called FE-Compatibility for any public key encryption scheme.

Definition 3. *A public key encryption scheme* $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *is said to be selective/adaptive FE-Compatible relative to a family of functions* $\mathcal{F}$ *if there exists two algorithms* $(\mathsf{FE.Keygen}, \mathsf{FE.Dec})$ *such that* $(\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.Keygen}, \mathsf{FE.Dec})$ *is a selectively/adaptively secure functional encryption scheme for the family* $\mathcal{F}$ *where:*

- $\mathsf{FE.Setup}(n) = \mathsf{PKE.Setup}(n)$. *In particular, if* $\mathsf{PKE.Setup}(n)$ *outputs* $(\mathsf{PK}, \mathsf{SK})$, *the output of* $\mathsf{FE.Setup}(n)$ *is* $(\mathsf{MPK} = \mathsf{PK}, \mathsf{MSK} = \mathsf{SK})$.
- $\mathsf{FE.Enc}(\mathsf{MPK}, \mathsf{m}) = \mathsf{PKE.Enc}(\mathsf{PK}, \mathsf{m})$.

Remark: Moreover, any such FE scheme is also compact because the size of the ciphertext is determined by the scheme PKE and doesn't depend on the size of the functions being queried.

4 An Impossibility Result

In this section, we will construct an IND-CCA secure encryption scheme that is not FE-Compatible according to Definition 3. Consider a function f_1 that on any input x of length $(n+1)$ bits, outputs the first n bits of x. Formally, we prove the following theorem:

Theorem 1. *Assuming the existence of lockable obfuscation, fully homomorphic encryption and IND-CCA secure public key encryption, the scheme* $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *described below is an IND-CCA secure public key encryption scheme that is not selective FE-Compatible even for a single function secret key query for any function family* $\mathcal{F}$ *such that* $f_1 \in \mathcal{F}$.

We know how to construct lockable obfuscation with perfect correctness from the learning with errors (LWE) assumption[GKW17,WZ17]. As a result, we get the following corollary:

Corollary 2 *Assuming LWE, fully homomorphic encryption and the existence of IND-CCA secure public key encryption, the scheme* $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *described below is an IND-CCA secure public key encryption scheme that is not selective FE-Compatible even for a single function secret key query for any function family* $\mathcal{F}$ *such that* $f_1 \in \mathcal{F}$.

Notation: Let the security parameter be n. Let $(\mathsf{Setup}_{\mathsf{CPA}}, \mathsf{Enc}_{\mathsf{CPA}}, \mathsf{Dec}_{\mathsf{CPA}})$ be an IND-CPA secure encryption scheme that encrypts 1 bit messages and produces ciphertexts of length $l_1(n)$, $(\mathsf{Setup}_{\mathsf{CCA}}, \mathsf{Enc}_{\mathsf{CCA}}, \mathsf{Dec}_{\mathsf{CCA}})$ be a CCA secure encryption scheme that encrypts messages of length $(n+1+l_1(n))$ and produces ciphertexts of size $l_2(n)$. Let $\mathsf{FHE} = (\mathsf{FHE.Setup}, \mathsf{FHE.Enc}, \mathsf{FHE.DecFHE.Eval})$ be a fully homomorphic encryption scheme that encrypts messages of length $(l_1(n)+l_2(n))$ and can evaluate any $\mathsf{Poly}(n)$-sized circuit. Let $(\mathcal{O}, \mathsf{Eval})$ be a secure lockable obfuscator for all $\mathsf{Poly}(n)$-sized circuits that take inputs of size $l_2(n)$ and produce outputs of size n. Our scheme $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ that encrypts messages of length $(n+1)$ is as follows:

- $\mathsf{PKE.Setup}(1^n)$**:**
  1. Compute $(\mathsf{PK}_{\mathsf{CPA}}, \mathsf{SK}_{\mathsf{CPA}}) \leftarrow \mathsf{Setup}_{\mathsf{CPA}}(1^n)$, $(\mathsf{PK}_{\mathsf{CCA}}, \mathsf{SK}_{\mathsf{CCA}}) \leftarrow \mathsf{Setup}_{\mathsf{CCA}}(1^n)$ and $(\mathsf{PK}_{\mathsf{FHE}}, \mathsf{SK}_{\mathsf{FHE}}) \leftarrow \mathsf{FHE.Setup}(1^n)$.
  2. Choose a random string $\alpha \in \{0,1\}^n$.
  3. Compute $\mathsf{CT}'_{\mathsf{CPA}} = \mathsf{Enc}_{\mathsf{CPA}}(\mathsf{PK}_{\mathsf{CPA}}, 0)$ and $\mathsf{CT}'_{\mathsf{CCA}} = \mathsf{Enc}_{\mathsf{CCA}}(\mathsf{PK}_{\mathsf{CCA}}, \alpha||0||\mathsf{CT}'_{\mathsf{CPA}})$. Let $\mathsf{CT}' = (\mathsf{CT}'_{\mathsf{CCA}}, \mathsf{CT}'_{\mathsf{CPA}})$. (In fact, CT' is an encryption of $(\alpha||0)$ using the encryption algorithm $\mathsf{PKE.Enc}$ described next).
  4. Compute $\mathsf{CT}'_{\mathsf{FHE}} = \mathsf{FHE.Enc}(\mathsf{PK}_{\mathsf{FHE}}, \mathsf{CT}')$.
  5. Generate $\tilde{P} = \mathcal{O}(n, P, \mathsf{SK}_{\mathsf{CPA}}, \alpha)$ using the tester program P described in Figure 1 where n is the security parameter, P is the program, $\mathsf{SK}_{\mathsf{CPA}}$ is the message and α is the lock value. In particular, the functionality of the obfuscated program $\tilde{P}$ is described in Figure 2. Note that Figure 2 is just for intuition and does not correspond to a formal specification.
  6. Output the public key as $\mathsf{PK} = (\mathsf{PK}_{\mathsf{CPA}}, \mathsf{PK}_{\mathsf{CCA}}, \mathsf{PK}_{\mathsf{FHE}}, \mathsf{CT}'_{\mathsf{FHE}}, \tilde{P})$. The secret key of the scheme is $\mathsf{SK} = \mathsf{SK}_{\mathsf{CCA}}$.
- $\mathsf{PKE.Enc}(\mathsf{PK}, \mathsf{m})$**:**
  1. Given an $(n+1)$ bit message m, let p be the last bit of m.
  2. Compute $\mathsf{CT}_{\mathsf{CPA}} = \mathsf{Enc}_{\mathsf{CPA}}(\mathsf{PK}_{\mathsf{CPA}}, p)$.
  3. Compute $\mathsf{CT}_{\mathsf{CCA}} = \mathsf{Enc}_{\mathsf{CCA}}(\mathsf{PK}_{\mathsf{CCA}}, \mathsf{m}||\mathsf{CT}_{\mathsf{CPA}})$.
  4. Output the ciphertext $\mathsf{CT} = (\mathsf{CT}_{\mathsf{CCA}}, \mathsf{CT}_{\mathsf{CPA}})$.
- $\mathsf{PKE.Dec}(\mathsf{SK}, \mathsf{CT})$**:**
  1. Parse $\mathsf{CT} = (\mathsf{CT}_{\mathsf{CCA}}, \mathsf{CT}_{\mathsf{CPA}})$. Recall that $\mathsf{SK} = \mathsf{SK}_{\mathsf{CCA}}$.
  2. Let $(\mathsf{m}||\mathsf{y}) = \mathsf{Dec}_{\mathsf{CCA}}(\mathsf{SK}_{\mathsf{CCA}}, \mathsf{CT}_{\mathsf{CCA}})$.
  3. If the above decryption outputs $\perp$ or if $y \neq \mathsf{CT}_{\mathsf{CPA}}$, output $\perp$.
  4. Else, output the message m.

```
                              Program P

Input : FHE ciphertext $\mathsf{CT}_{\mathsf{FHE}}$
Constants : $\mathsf{SK}_{\mathsf{FHE}}$

 1. Output $\mathsf{FHE.Dec}(\mathsf{SK}_{\mathsf{FHE}}, \mathsf{CT}_{\mathsf{FHE}})$.
```

Fig. 1: Tester Program (as in lockable obfuscation notation)

```
Program $\tilde{P}$

Input : FHE ciphertext $\mathsf{CT}_{\mathsf{FHE}}$
Constants : $\mathsf{SK}_{\mathsf{FHE}}, \alpha, \mathsf{SK}_{\mathsf{CPA}}$

1. Compute $\mathsf{y} \leftarrow \mathsf{FHE.Dec}(\mathsf{SK}_{\mathsf{FHE}}, \mathsf{CT}_{\mathsf{FHE}})$.
2. If $\mathsf{y} = \alpha$, output $\mathsf{SK}_{\mathsf{CPA}}$. Else, output $\perp$.
```

Fig. 2: Functionality of lockable obfuscated tester program

We now prove Theorem 1.

Correctness: It can be observed that if the schemes $(\mathsf{Setup}_{\mathsf{CPA}}, \mathsf{Enc}_{\mathsf{CPA}}, \mathsf{Dec}_{\mathsf{CPA}})$ and $(\mathsf{Setup}_{\mathsf{CCA}}, \mathsf{Enc}_{\mathsf{CCA}}, \mathsf{Dec}_{\mathsf{CCA}})$ are correct except with negligible probability, then PKE is correct except with negligible probability. That is, $\mathsf{PKE.Dec}(\mathsf{PKE.Enc}(\mathsf{PK}, \mathsf{m}), \mathsf{SK}) = \mathsf{m}$ for any message $\mathsf{m} \in \{0,1\}^{(n+1)}$.

To prove our theorem we need to show two things. First, we will show that any candidate functional encryption scheme that includes a "all but last bit reveal" functionality which shares the setup and encrypt algorithms with the above public key encryption scheme must be insecure. Second, we show that the scheme PKE actually does have IND-CCA security under certain assumptions. Putting these together will yield our theorem.

4.1 An Attack

In this section, assuming the correctness of the encryption schemes used and correctness of the obfuscator, we show that the above scheme PKE is not FE-Compatible. Suppose it is indeed FE-Compatible. We will arrive at a contradiction. Formally, we prove the following lemma.

Lemma 1. *Any scheme* $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.Keygen}, \mathsf{FE.Dec})$ *where* $\mathsf{FE.Setup}(.) = \mathsf{PKE.Setup}(.)$ *and* $\mathsf{FE.Enc}(.) = \mathsf{PKE.Enc}(.)$ *is not selectively secure even for just 1 function secret key query for any function family* $\mathcal{F}$ *such that* $f_1 \in \mathcal{F}$.

Proof. Consider a FE adversary $\mathcal{A}$ who interacts with a FE challenger in the selective IND-security game as follows:

1. In the first round, $\mathcal{A}$ submits two messages $\mathsf{m}_0 = (0^n||0)$ and $\mathsf{m}_1 = (0^n||1)$.
2. $\mathcal{A}$ asks for a function secret key corresponding to the following function f_1: on input x of length $(n+1)$ bits, $f_1(x)$ outputs the first n bits of x. Note that since the first n bits of m_0 and m_1 are equal, this is a valid function secret key query.

3. The challenger runs the setup algorithm and generates PK, SK. He gives PK to the adversary along with the function secret key SK_{f_1}. Also, the challenger picks a bit b at random and sends $\mathsf{CT}^* = \mathsf{PKE.Enc}(\mathsf{PK}, \mathsf{m}_b)$.
4. Let the challenge ciphertext be $\mathsf{CT}^* = (\mathsf{CT}^*_{\mathsf{CCA}}, \mathsf{CT}^*_{\mathsf{CPA}})$. The adversary computes a FHE ciphertext $\mathsf{CT}_{\mathsf{FHE}} = \mathsf{FHE.Eval}(\mathsf{FE.Dec}(\mathsf{SK}_f, \cdot), \mathsf{CT}'_{\mathsf{FHE}})$ using the ciphertext $\mathsf{CT}'_{\mathsf{FHE}}$ in the public key and the function secret key SK_f. $\mathcal{A}$ then runs the obfuscated program $\tilde{P}$ on input $\mathsf{CT}_{\mathsf{FHE}}$. That is, run $\mathsf{Eval}(\tilde{P}, \mathsf{CT}_{\mathsf{FHE}})$ to receive output $\mathsf{SK}'_{\mathsf{CPA}}$. It then computes $b' = \mathsf{Dec}_{\mathsf{CPA}}(\mathsf{SK}'_{\mathsf{CPA}}, \mathsf{CT}^*_{\mathsf{CPA}})$ and outputs b' to the challenger.

Analysis: We now show why the adversary's guess b' is equal to the challenger's random bit b except with negligible probability. From the correctness of the FE scheme, SK_{f_1} must be a correct function secret key for the function f_1. First, from the correctness of the FHE scheme, observe that $\mathsf{CT}_{\mathsf{FHE}} = \mathsf{FHE.Eval}(\mathsf{SK}_f, \mathsf{CT}'_{\mathsf{FHE}})$ is an encryption of the random string α using the algorithm $\mathsf{FHE.Enc}$. Now, notice that when this ciphertext $\mathsf{CT}_{\mathsf{FHE}}$ is a correct encryption of α. So, when it is fed as input to the program $\tilde{P}$, from the correctness of lockable obfuscation, the program outputs the secret key of the IND-CPA secure encryption scheme - $\mathsf{SK}_{\mathsf{CPA}}$ (which we denoted as $\mathsf{SK}'_{\mathsf{CPA}}$). Therefore, now the adversary's strategy easily follows. $\mathcal{A}$ uses $\mathsf{SK}_{\mathsf{CPA}}$ to decrypt $\mathsf{CT}^*_{\mathsf{CPA}}$ and from the correctness of the IND-CPA secure encryption scheme, this decrypts to give the value b correctly, which is the adversary's output.

Hence, the adversary can break the selective IND-security of the FE scheme which is a contradiction. Note that the negligible error comes from the fact that the IND-CPA secure encryption scheme, the FE scheme, the lockable obfuscation scheme and the FHE scheme are all correct except with negligible probability.

4.2 IND-CCA Security

We now prove that the scheme is IND-CCA secure. Our proof strategy is organized along the lines around detecting a bad query event which follows the work of Myers and Shelat[MS09] and Hohenberger, Lewko and Waters[HLW12] who proved multibit CCA security from the existence of 1-bit CCA security. Formally, we prove the following lemma:

Lemma 2. *Assuming the hardness of learning with errors (LWE),* $(\mathsf{Setup}_{\mathsf{CPA}}, \mathsf{Enc}_{\mathsf{CPA}}, \mathsf{Dec}_{\mathsf{CPA}})$ *is an IND-CPA secure public key encryption scheme and* $(\mathsf{Setup}_{\mathsf{CCA}}, \mathsf{Enc}_{\mathsf{CCA}}, \mathsf{Dec}_{\mathsf{CCA}})$ *is an IND-CCA secure public key encryption scheme,* $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *is an IND-CCA secure public key encryption scheme.*

Proof. We begin our proof by defining a "Bad-Query" event that is defined within the context of the attacker playing the IND-CCA security game on the encryption scheme PKE.

Definition 4. ***(Bad Query Event):*** *Let* PK *be the public key of the scheme* PKE *that is given to the adversary. We say that a bad query event has occurred* *during an execution of the IND-CCA security game between the adversary $\mathcal{A}$ and the challenger if $\mathcal{A}$ makes a decryption query of the form $\mathsf{CT} = (\mathsf{CT}_1, \mathsf{CT}_2)$ such that $\mathsf{CT}_1 = \mathsf{CT}'_{\mathsf{CCA}}$, where $\mathsf{CT}'_{\mathsf{CCA}}$ was created by the setup algorithm $\mathsf{PKE.Setup}$.*

In order to prove IND-CCA security of our scheme, we will rely on the following claim :

Claim. A Bad Query Event does not take place except with negligible probability in n, where the probability is taken over the coins of the adversary and the challenger playing the IND-CCA security game.

We defer the proof of this claim to the next section. Here, we show that our scheme is IND-CCA secure assuming the claim holds true. We will prove this via a series of hybrid experiments where we show that every successive pair of hybrids is computationally indistinguishable and the final hybrid is independent of the challenge bit b and hence the attacker's advantage will be 0 in the final hybrid.

- Hyb_1**:** This is the real world experiment with challenge bit b chosen randomly. The challenge ciphertext is $\mathsf{CT}^* = (\mathsf{CT}^*_{\mathsf{CCA}}, \mathsf{CT}^*_{\mathsf{CPA}})$.
- Hyb_2**:** This hybrid is identical to the previous hybrid except that now, the decryption oracle rejects[9] for any ciphertext query $\mathsf{CT} = (\mathsf{CT}_1, \mathsf{CT}_2)$ if $\mathsf{CT}_1 = \mathsf{CT}'_{\mathsf{CCA}}$. Note that the oracle also continues to reject the challenge ciphertext as before.
- Hyb_3**:** This hybrid is identical to the previous hybrid except that during setup, $\mathsf{CT}'_{\mathsf{CCA}}$ is now computed as $\mathsf{CT}'_{\mathsf{CCA}} = \mathsf{Enc}_{\mathsf{CCA}}(\mathsf{PK}_{\mathsf{CCA}}, 0^{n+1} || \mathsf{CT}'_{\mathsf{CPA}})$.
- Hyb_4**:** This hybrid is identical to the previous hybrid except that in the public key, $\tilde{P}$ is replaced with the simulated obfuscated program - i.e $\mathsf{Sim}(n, 1^{|P|}, 1^{|\mathsf{SK}_{\mathsf{CPA}}|})$ where Sim is the simulator of the lockable obfuscation scheme.
- Hyb_5**:** This hybrid is identical to the previous hybrid except that in the challenge ciphertext $\mathsf{CT}^* = (\mathsf{CT}^*_{\mathsf{CCA}}, \mathsf{CT}^*_{\mathsf{CPA}})$, $\mathsf{CT}^*_{\mathsf{CPA}}$ is now computed independent of the bit b as follows: $\mathsf{CT}^*_{\mathsf{CPA}} = \mathsf{Enc}_{\mathsf{CPA}}(\mathsf{PK}_{\mathsf{CPA}}, 0)$.
- Hyb_6**:** This hybrid is identical to the previous hybrid except that now, the decryption oracle also rejects any ciphertext query $\mathsf{CT} = (\mathsf{CT}_1, \mathsf{CT}_2)$ if $\mathsf{CT}_1 = \mathsf{CT}^*_{\mathsf{CCA}}$.
- Hyb_7**:** This hybrid is identical to the previous hybrid except that in the challenge ciphertext, $\mathsf{CT}^*_{\mathsf{CCA}}$ is now computed independent of the bit b as follows: $\mathsf{CT}^*_{\mathsf{CCA}} = \mathsf{Enc}_{\mathsf{CPA}}(\mathsf{PK}_{\mathsf{CCA}}, 0^{n+1} || \mathsf{CT}^*_{\mathsf{CPA}})$.

Observe that in this last hybrid, the challenge ciphertext is created independent of the bit b. Hence, the attacker's advantage in this hybrid is negligible.

We will now show the indistinguishability of every successive pair of hybrids.

Claim. Assuming Claim 4.2 holds, Hyb_1 is computationally indistinguishable from Hyb_2.

[9] Throughout the paper, we use rejecting an input and producing output $\perp$ for the input interchangeably.

Proof. The only difference between the two hybrids is that in Hyb_2, the decryption oracle rejects queries of the form $\mathsf{CT} = (\mathsf{CT}_1, \mathsf{CT}_2)$ where $\mathsf{CT}_1 = \mathsf{CT}'_{\mathsf{CCA}}$ while such queries are not rejected by the oracle in Hyb_1. However, Claim 4.2 essentially proves that such queries (which we have defined as the occurrence of a bad query event) are never made by the adversary except with negligible probability. Therefore, if Claim 4.2 holds, Hyb_1 is computationally indistinguishable from Hyb_2.

Claim. Assuming that $(\mathsf{Setup}_{\mathsf{CCA}}, \mathsf{Enc}_{\mathsf{CCA}}, \mathsf{Dec}_{\mathsf{CCA}})$ is an IND-CCA secure encryption scheme, Hyb_2 is computationally indistinguishable from Hyb_3.

Proof. The only difference is that in Hyb_2, $\mathsf{CT}'_{\mathsf{CCA}} = \mathsf{Enc}_{\mathsf{CCA}}(\mathsf{PK}_{\mathsf{CCA}}, \alpha||0||\mathsf{CT}'_{\mathsf{CPA}})$ while in Hyb_3, $\mathsf{CT}'_{\mathsf{CCA}} = \mathsf{Enc}_{\mathsf{CCA}}(\mathsf{PK}_{\mathsf{CCA}}, 0^{n+1}||\mathsf{CT}'_{\mathsf{CPA}})$. We can show that if there exists an adversary $\mathcal{A}$ that can distinguish between these two hybrids, there exists an adversary $\mathcal{B}$ that can break the CCA security of the encryption scheme $(\mathsf{Setup}_{\mathsf{CCA}}, \mathsf{Enc}_{\mathsf{CCA}}, \mathsf{Dec}_{\mathsf{CCA}})$. We defer the details of the proof to the full version.

Claim. Assuming that $(\mathcal{O}, \mathsf{Eval})$ is a secure lockable obfuscator, Hyb_3 is computationally indistinguishable from Hyb_4.

Proof. The only difference between the two hybrids is that in Hyb_3, the public key contains $\mathcal{O}(n, P, \mathsf{SK}_{\mathsf{CPA}}, \alpha)$ while in Hyb_4, it contains the simulated program - $\mathsf{Sim}(n, 1^{|P|}, 1^{|\mathsf{SK}_{\mathsf{CPA}}|})$. Since α is picked uniformly at random and is used only as the lock value in the obfuscated program and nowhere else, from the security of lockable obfuscation, the two hybrids will be computationally indistinguishable. We now describe the reduction.

Consider an adversary $\mathcal{A}$ that can distinguish between these two hybrids. We will now design a reduction $\mathcal{A}_{\mathsf{lock}}$ that uses $\mathcal{A}$ to break the security of the lockable obfuscation scheme. $\mathcal{A}_{\mathsf{lock}}$ interacts with $\mathcal{A}$ and runs the experiment exactly as in Hyb_3 except generating the obfuscated program. $\mathcal{A}_{\mathsf{lock}}$ interacts with a challenger $\mathcal{C}$ for the lockable obfuscation scheme. $\mathcal{A}_{\mathsf{lock}}$ sends the program P and the message $\mathsf{SK}_{\mathsf{CPA}}$ to the challenger $\mathcal{C}$. $\mathcal{C}$ sends back either $\mathcal{O}(n, P, \mathsf{SK}_{\mathsf{CPA}}, \alpha)$ where α is picked uniformly at random or a simulated obfuscated circuit $\mathsf{Sim}(n, 1^|P|, 1^|\mathsf{SK}_{\mathsf{CPA}}|)$. $\mathcal{A}_{\mathsf{lock}}$ sets this as the obfuscated circuit $\tilde{P}$ and continues with the experiment as in Hyb_3. Now, it easily follows that if $\mathcal{A}$ can distinguish between the two hybrids, $\mathcal{A}_{\mathsf{lock}}$ can use the same distinguishing guess to break the security of the lockable obfuscation scheme which is a contradiction.

Claim. Assuming that $(\mathsf{Setup}_{\mathsf{CPA}}, \mathsf{Enc}_{\mathsf{CPA}}, \mathsf{Dec}_{\mathsf{CPA}})$ is an IND-CPA secure encryption scheme, Hyb_4 is computationally indistinguishable from Hyb_5.

Proof. The only difference between the two hybrids is in the challenge ciphertexts. In Hyb_4, $\mathsf{CT}^*_{\mathsf{CPA}} = \mathsf{Enc}_{\mathsf{CPA}}(\mathsf{PK}_{\mathsf{CPA}}, p_b)$ while in Hyb_5, $\mathsf{CT}^*_{\mathsf{CPA}} = \mathsf{Enc}_{\mathsf{CCA}}(\mathsf{PK}_{\mathsf{CPA}}, 0)$. Here, p is the last bit of the message m_b. We can show that if there exists an adversary $\mathcal{A}$ that can distinguish between these two hybrids, there exists an adversary $\mathcal{B}$ that can break the CPA security of the encryption scheme $(\mathsf{Setup}_{\mathsf{CPA}}, \mathsf{Enc}_{\mathsf{CPA}}, \mathsf{Dec}_{\mathsf{CPA}})$. We defer the details of the proof to the full version.

Claim. Hyb_5 is identical to Hyb_6.

Proof. The only difference between the two hybrids is that in Hyb_6, the decryption oracle rejects any ciphertext query $\mathsf{CT} = (\mathsf{CT}_1, \mathsf{CT}_2)$ if $\mathsf{CT}_1 = \mathsf{CT}^*_{\mathsf{CCA}}$. First, observe that if $\mathsf{CT}_2 = \mathsf{CT}^*_{\mathsf{CPA}}$, then CT is in fact the challenge ciphertext CT^* itself and hence even Hyb_6 would reject the query. On the other hand, if $\mathsf{CT}_2 \neq \mathsf{CT}^*_{\mathsf{CPA}}$ but $\mathsf{CT}_1 = \mathsf{CT}^*_{\mathsf{CCA}}$, then, $\mathsf{Dec}_{\mathsf{CCA}}(\mathsf{SK}_{\mathsf{CCA}}, \mathsf{CT}_1)$ produces (m^*, y^*) such that $y^* \neq \mathsf{CT}_2$. This is because y^* would in fact be equal to $\mathsf{CT}^*_{\mathsf{CPA}}$. Hence, even Hyb_5 would reject these queries and so the two hybrids are identical.

Claim. Assuming that $(\mathsf{Setup}_{\mathsf{CCA}}, \mathsf{Enc}_{\mathsf{CCA}}, \mathsf{Dec}_{\mathsf{CCA}})$ is an IND-CCA secure encryption scheme, Hyb_6 is computationally indistinguishable from Hyb_7.

Proof. The only difference between the two hybrids is in the challenge ciphertext $\mathsf{CT}^* = (\mathsf{CT}^*_{\mathsf{CCA}}, \mathsf{CT}^*_{\mathsf{CPA}})$. In Hyb_6, $\mathsf{CT}^*_{\mathsf{CCA}} = \mathsf{Enc}_{\mathsf{CCA}}(\mathsf{PK}_{\mathsf{CCA}}, \mathsf{m}_b || \mathsf{CT}^*_{\mathsf{CPA}})$ while in Hyb_7, $\mathsf{CT}^*_{\mathsf{CCA}} = \mathsf{Enc}_{\mathsf{CCA}}(\mathsf{PK}_{\mathsf{CCA}}, 0^{n+1} || \mathsf{CT}^*_{\mathsf{CPA}})$. We can show that if there exists an adversary $\mathcal{A}$ that can distinguish between these two hybrids, there exists an adversary $\mathcal{B}$ that can break the CCA security of the encryption scheme $(\mathsf{Setup}_{\mathsf{CCA}}, \mathsf{Enc}_{\mathsf{CCA}}, \mathsf{Dec}_{\mathsf{CCA}})$. We defer the details of the proof to the full version.

4.3 Proof of Claim 4.2

Instead of proving the claim directly, we first prove it for an alternate IND-CCA security game and then show how it holds even in the actual IND-CCA security game.

Alternate IND-CCA Game. This is same as the original game except that the Challenger now computes $\mathsf{CT}'_{\mathsf{CCA}}$ during setup as follows: $\mathsf{CT}'_{\mathsf{CCA}} = \mathsf{Enc}_{\mathsf{CCA}}(\mathsf{PK}_{\mathsf{CCA}}, 0^{n+1} || \mathsf{CT}'_{\mathsf{CPA}})$. That is, α is no longer part of the message being encrypted. For this alternate IND-CCA game, the Bad Query Event remains the same: i.e, the event occurs if the adversary makes a query $\mathsf{CT} = (\mathsf{CT}_1, \mathsf{CT}_2)$ to the decryption oracle where $\mathsf{CT}_1 = \mathsf{CT}'_{\mathsf{CCA}}$. Now, via a sequence of hybrids, we show that Claim 4.2 holds for this alternate IND-CCA game. That is, we show that the Bad Query Event happens with negligible probability.

– Hyb_1**:** This hybrid corresponds to the alternate IND-CCA game as described above.
– Hyb_2**:** This hybrid is identical to the previous hybrid except that in the public key, $\tilde{P}$ is replaced with the simulated obfuscated program - i.e $\mathsf{Sim}(n, 1^{|P|}, 1^{|\mathsf{SK}_{\mathsf{CPA}}|})$ where Sim is the simulator of the lockable obfuscation scheme.
– Hyb_3**:** This hybrid is identical to the previous hybrid except that the ciphertext $\mathsf{CT}'_{\mathsf{FHE}}$ is now computed as $\mathsf{CT}'_{\mathsf{FHE}} = \mathsf{FHE.Enc}(\mathsf{PK}_{\mathsf{FHE}}, 0^{l_1(n)+l_2(n)})$.

We now show that every successive pair of hybrids is computationally indistinguishable. This proves that the probability that the Bad Query Event occurs is the same for every pair of successive hybrids. Finally, we show that in the last hybrid Hyb_4, the probability that the Bad Query Event occurs is negligible.

Claim. Assuming that $(\mathcal{O}, \mathsf{Eval})$ is a secure lockable obfuscator, Hyb_1 is computationally indistinguishable from Hyb_2.

Proof. The proof is same as the proof of Claim 4.2.

Claim. Assuming that $(\mathsf{FHE}.\mathit{setup}, \mathsf{FHE}.\mathit{enc}, \mathsf{FHE}.\mathit{dec})$ is an IND-CPA secure fully homomorphic encryption scheme, Hyb_2 is computationally indistinguishable from Hyb_3.

Proof. The proof is same as the proof of Claim 4.2.

Claim. $\Pr[\text{Bad Query Event occurs in } \mathsf{Hyb}_3] = \text{negligible}(n)$.

Proof. This is because the ciphertext $\mathsf{CT}'_{\mathsf{CCA}}$ does not appear at all in the public key anymore! Even if the adversary knew the value of $(\mathsf{CT}'_{\mathsf{CPA}})$, the only information that the adversary has about $\mathsf{CT}'_{\mathsf{CCA}}$ is that it is an encryption of $(0^{n+1}||\mathsf{CT}'_{\mathsf{CPA}})$ using public key $\mathsf{PK}_{\mathsf{CCA}}$.

First, observe that the number of possible ciphertexts for the message $(0^{n+1}|| \mathsf{CT}'_{\mathsf{CPA}})$ must be at least super-polynomial in n. This follows from the CPA security of the encryption scheme $(\mathsf{Setup}_{\mathsf{CCA}}, \mathsf{Enc}_{\mathsf{CCA}}, \mathsf{Dec}_{\mathsf{CCA}})$ because if this wasn't true, a polynomial time adversary can break the CPA security by generating all possible ciphertexts for $(0^{n+1}||\mathsf{CT}'_{\mathsf{CPA}})$ and testing it with the challenge ciphertext.

Now, notice that to make the Bad Query Event occur, the adversary will just have to guess the value of $\mathsf{CT}'_{\mathsf{CCA}}$ (or the randomness that was used in the encryption to generate $\mathsf{CT}'_{\mathsf{CCA}}$) and this can be done only with negligible probability.

Original IND-CCA Game. We show that the Bad Query Event happens only with negligible probability even in the original IND-CCA game. Formally, we prove the following lemma:

Lemma 3. *Assuming* $(\mathsf{Setup}_{\mathsf{CCA}}, \mathsf{Enc}_{\mathsf{CCA}}, \mathsf{Dec}_{\mathsf{CCA}})$ *is a CCA secure encryption scheme and that the Bad Query Event does not occur in the Alternate CCA game described above except with negligible probability, the Bad Query Event does not occur in the original CCA security game for the encryption scheme* PKE *except with negligible probability.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that makes the Bad Query Event occur with non-negligible probability. We now construct an algorithm $\mathcal{B}$ that breaks the IND-CCA security of $(\mathsf{Setup}_{\mathsf{CCA}}, \mathsf{Enc}_{\mathsf{CCA}}, \mathsf{Dec}_{\mathsf{CCA}})$. $\mathcal{B}$ acts as the challenger of the IND-CCA security game for the scheme PKE in its interaction with $\mathcal{A}$. First, $\mathcal{B}$ interacts with its challenger and receives the public key $\mathsf{PK}_{\mathsf{CCA}}$. $\mathcal{B}$ then runs the setup algorithm $\mathsf{PKE.Setup}$,(except the $\mathsf{Setup}_{\mathsf{CCA}}$ part) to compute the public keys $\mathsf{PK}_{\mathsf{CPA}}, \mathsf{PK}_{\mathsf{FHE}}$. It computes $\mathsf{CT}'_{\mathsf{CPA}}$ as done by the setup algorithm. $\mathcal{B}$ then sends the pair $(\alpha||0||\mathsf{CT}'_{\mathsf{CPA}}, 0^{n+1}||\mathsf{CT}'_{\mathsf{CPA}})$ as the two challenge messages to the challenger and sets the response as $\mathsf{CT}'_{\mathsf{CCA}}$. $\mathcal{B}$ continues with the rest of the game acting as the challenger to $\mathcal{A}$. Whenever $\mathcal{A}$ makes a decryption query

$(\mathsf{CT}_1, \mathsf{CT}_2)$, if $\mathsf{CT}_1 \neq \mathsf{CT}'_{\mathsf{CCA}}$, it queries the decryption oracle of its challenger with CT_1 and uses this to respond to $\mathcal{A}$ as done in the original game. Similarly, $\mathcal{B}$ also creates the challenge ciphertext. If $\mathcal{B}$ ever receives a query $(\mathsf{CT}_1, \mathsf{CT}_2)$ from $\mathcal{A}$ to the decryption oracle such that $\mathsf{CT}_1 = \mathsf{CT}'_{\mathsf{CCA}}$, it immediately halts the game with $\mathcal{A}$ and outputs the guess 0 to its challenger. If such a query never happens, it outputs 1 to the challenger after completing the game with $\mathcal{A}$.

We now analyze why this works. The algorithm $\mathcal{B}$ knows that if its challenger gave an encryption of 0^{n+1}, then its interaction with $\mathcal{A}$ corresponds to the alternate IND-CCA game described earlier. Here, we know that the adversary $\mathcal{A}$ can not make the Bad Query Event occur. Therefore, if the adversary $\mathcal{A}$ makes the Bad Query Event occur, then it must occur in the case that $\mathsf{CT}'_{\mathsf{CCA}}$ is an encryption of $(\alpha||0||\mathsf{CT}'_{\mathsf{CPA}})$. Hence, $\mathcal{B}$ guesses 0 in that case. On the other hand, if the adversary $\mathcal{A}$ does not make the Bad Query Event occur, then it must be the case that 0^{n+1} was encrypted. This is because we assumed that $\mathcal{A}$ can make the Bad Query Event occur with non-negligible probability in the original IND-CCA security game. This completes the proof.

Note that the reduction is actually not interested in completing the game with $\mathcal{A}$ in the event that $\mathcal{B}$ halts. That is, $\mathcal{B}$ does not care whether $\mathcal{A}$ wins the IND-CCA game but is rather more interested in whether $\mathcal{A}$ makes a Bad Query Event occur.

Remark: At first glance, there seems to be a circularity issue in trying to prove IND-CCA security of our scheme. That is, in order to prove indistinguishability of the main hybrids, we require to first erase α which depends on no queries being made to the decryption oracle that contain $\mathsf{CT}'_{\mathsf{CCA}}$. On the other hand, it seems difficult to directly argue that no such queries are made because of the presence of α in $\mathsf{CT}'_{\mathsf{CCA}}$. This causes a circularity. We get around this issue using the alternate IND-CCA game where α is erased. In this game, we show that the bad query event can't occur and then using a reduction to the underlying encryption scheme's security, we can eventually show that the bad query event does not occur even in the original CCA security game.

This technique is very similar to [MS09,HLW12]. In these works, they construct CCA secure encryption and in the process, they run into a similar circularity issue. The analog of α was the randomness used for encryption and this randomness is in fact encrypted by an inner encryption scheme.

This completes the proof of Theorem 1.

5 Building FE-Compatible Encryption Schemes

We first define a new notion called puncturable tag based encryption[10]. In the next subsection, we show how to construct a selective IND-CCA secure public key encryption scheme from any puncturable tag based encryption scheme. We

[10] Previously, [MH14] also introduced a primitive called puncturable tag based encryption which is completely different from the one we define here.

call such a selective IND-CCA secure public key encryption scheme as "Special-CCA". In the following subsection, we show how to instantiate a "Special-CCA" secure encryption scheme with several existing popular encryption schemes in literature. Finally, we show that this "Special-CCA" secure public key encryption scheme is FE-Compatible.

5.1 Puncturable Tag Based Encryption

In this section, we define a new primitive called puncturable tag based encryption (PTBE) that is a modification of tag based encryption schemes [Kil06] but with two more algorithms. We then show how several well known encryption schemes in literature (based on various assumptions) do in fact fit into the framework of puncturable tag based encryption.

Let n denote the security parameter and $\mathcal{X} = \{\mathcal{X}_n\}_{n\in\mathbf{N}}$, $\mathcal{T} = \{\mathcal{T}_n\}_{n\in\mathbf{N}}$ denote ensembles where each $\mathcal{X}_n$ and $\mathcal{T}_n$ is a finite set. Formally, a puncturable tag based encryption scheme $\mathsf{PTBE} = (\mathsf{PTBE.Setup}, \mathsf{PTBE.Enc}, \mathsf{PTBE.Dec}, \mathsf{PTBE.Setup\text{-}Alt}, \mathsf{PTBE.Setup\text{-}Alt\text{-}1}, \mathsf{PTBE.Dec\text{-}Alt})$ consists of the following algorithms:

- $\mathsf{PTBE.Setup}(1^n)$:
  Given the security parameter n, it generates a public key PK and a secret key SK.
- $\mathsf{PTBE.Enc}(\mathsf{PK}, \mathsf{t}, \mathsf{m})$:
  Given a message $\mathsf{m} \in \mathcal{X}_n$, a tag $\mathsf{t} \in \mathcal{T}_n$ and the public key PK as input, the encryption algorithm outputs a ciphertext CT.
- $\mathsf{PTBE.Dec}(\mathsf{SK}, \mathsf{t}, \mathsf{CT})$:
  Given a ciphertext CT, a tag $\mathsf{t} \in \mathcal{T}_n$ and the secret key SK as input, the decryption algorithm outputs a string $y \in \mathcal{X}_n$ or $\perp$.
- $\mathsf{PTBE.Setup\text{-}Alt}(1^n, \mathsf{t}^*, \mathsf{m}^*)$:
  Given the security parameter n, a tag $\mathsf{t}^* \in \mathcal{T}_n$ and a message $\mathsf{m}^* \in \mathcal{X}_n$, it generates a public key PK, a secret key SK, an alternate secret key $\mathsf{SK\text{-}Alt}$ and a ciphertext CT^*.
- $\mathsf{PTBE.Setup\text{-}Alt\text{-}1}(1^n, \mathsf{t}^*, \mathsf{m}^*)$:
  Given the security parameter n, a tag $\mathsf{t}^* \in \mathcal{T}_n$ and a message $\mathsf{m}^* \in \mathcal{X}_n$, it generates a public key PK, a secret key SK, an alternate secret key $\mathsf{SK\text{-}Alt}$ and a ciphertext CT^*.
- $\mathsf{PTBE.Dec\text{-}Alt}(\mathsf{SK\text{-}Alt}, \mathsf{t}, \mathsf{CT})$:
  Given a ciphertext CT, a tag $\mathsf{t} \in \mathcal{T}_n$ and an alternate secret key $\mathsf{SK\text{-}Alt}$ as input, the alternate decryption algorithm outputs a string $y \in \mathcal{X}_n$ or $\perp$.

Remark: For technical reasons, to make our proofs simpler while instantiating our "Special-CCA" secure encryption schemes, we use two setup-alt algorithms (that albeit perform a very similar role). We provide more details about this in a remark at the end of Section **??**. Alternatively, we could just use one setup-alt algorithm in the abstraction and make the proof a bit more complicated. We choose the former option in this writeup.

Correctness: A puncturable tag based encryption scheme PTBE is correct if for all messages $\mathsf{m} \in \mathcal{X}_n$ and all tags $\mathsf{t} \in \mathcal{T}_n$

$$\Pr\left[\begin{array}{c}(\mathsf{PK},\mathsf{SK}) \leftarrow \mathsf{PTBE.Setup}(1^{\mathsf{n}})\\ \mathsf{PTBE.Dec}(\mathsf{SK},\mathsf{t},\mathsf{PTBE.Enc}(\mathsf{PK},\mathsf{t},\mathsf{m})) = \mathsf{m}\end{array}\right] = 1$$

The probability is over the randomness used in the setup, encryption and decryption algorithms.

For security, we require the following four properties:

1. **Equivalent on all but challenge tag:** For any message $\mathsf{m}^* \in \mathcal{X}_n$, any tag $\mathsf{t}^* \in \mathcal{T}_n$, for all ciphertexts CT and all tags $\mathsf{t} \in \mathcal{T}_n$ such that $\mathsf{t} \neq \mathsf{t}^*$, we require that:

$$\Pr\left[\begin{array}{c}(\mathsf{PK},\mathsf{SK},\mathsf{SK\text{-}Alt},\mathsf{CT}^*)\leftarrow \mathsf{PTBE.Setup\text{-}Alt}(1^{\mathsf{n}},\mathsf{t}^*,\mathsf{m}^*)\\ \mathsf{PTBE.Dec}(\mathsf{SK},\mathsf{t},\mathsf{CT}) = \mathsf{PTBE.Dec\text{-}Alt}(\mathsf{SK\text{-}Alt},\mathsf{t},\mathsf{CT})\end{array}\right] = 1$$

   The probability is over the randomness used in all the above algorithms.
2. **Indistinguishability of parameters:** The output of the following two experiments must be computationally indistinguishable for all messages m^* and tags t^*:
   (a) **Experiment 1:**
   Run $\mathsf{PTBE.Setup}(1^n)$ to generate $(\mathsf{PK},\mathsf{SK})$. Compute $\mathsf{CT}^* = \mathsf{PTBE.Enc}(\mathsf{PK},\mathsf{t}^*,\mathsf{m}^*)$ and output $(\mathsf{PK},\mathsf{SK},\mathsf{CT}^*)$.
   (b) **Experiment 2:**
   Run $\mathsf{PTBE.Setup\text{-}Alt}(1^n,\mathsf{t}^*,\mathsf{m}^*)$ to generate $(\mathsf{PK},\mathsf{SK},\mathsf{SK\text{-}Alt},\mathsf{CT}^*)$ and output $(\mathsf{PK},\mathsf{SK},\mathsf{CT}^*)$.
3. **Indistinguishability of alternate setups:** The output of the following two experiments must be indistinguishable for all messages m^* and tags t^*:
   (a) **Experiment 1:**
   Run $\mathsf{PTBE.Setup\text{-}Alt}(1^n,\mathsf{t}^*,\mathsf{m}^*)$ to generate $(\mathsf{PK},\mathsf{SK},\mathsf{SK\text{-}Alt},\mathsf{CT}^*)$ and output $(\mathsf{PK},\mathsf{SK\text{-}Alt},\mathsf{CT}^*)$.
   (b) **Experiment 2:**
   Run $\mathsf{PTBE.Setup\text{-}Alt\text{-}1}(1^n,\mathsf{t}^*,\mathsf{m}^*)$ to generate $(\mathsf{PK},\mathsf{SK},\mathsf{SK\text{-}Alt},\mathsf{CT}^*)$ and output $(\mathsf{PK},\mathsf{SK\text{-}Alt},\mathsf{CT}^*)$.
4. **Indistinguishability of messages:** For this property to hold, we require the adversary's advantage to be negligible in the following game between an adversary $\mathcal{A}$ and a challenger Ch:
   (a) $\mathcal{A}$ sends $(\mathsf{t}^*,\mathsf{m}_0^*,\mathsf{m}_1^*)$ to the challenger.
   (b) Ch chooses a random bit b and runs $\mathsf{PTBE.Setup\text{-}Alt\text{-}1}(1^n,\mathsf{t}^*,\mathsf{m}_b^*)$ to generate $(\mathsf{PK},\mathsf{SK\text{-}Alt},\mathsf{CT}^*)$ and gives the adversary $(\mathsf{PK},\mathsf{SK\text{-}Alt},\mathsf{CT}^*)$.
   (c) $\mathcal{A}$ submits a guess b' and wins if $b' = b$. The adversary's advantage in this game is defined to be $2 * |\mathsf{Pr}[\mathsf{b} = \mathsf{b}'] - 1/2|$.

5.2 Special-CCA secure encryption scheme

In this section, we show how to build a selective CCA secure encryption scheme from any PTBE with the addition of one time signatures. Recall that we define selective CCA secure encryption schemes in the full version. We call such a CCA secure encryption scheme as "Special-CCA". Formally, we prove the following theorem:

Theorem 3. *Given a puncturable tag based encryption scheme* $\mathsf{PTBE} = (\mathsf{PTBE}$ $.\mathsf{Setup}, \mathsf{PTBE.Enc}, \mathsf{PTBE.Dec}, \mathsf{PTBE.Setup\text{-}Alt}, \mathsf{PTBE.Setup\text{-}Alt\text{-}1}, \mathsf{PTBE.Dec\text{-}Alt})$ *and a strongly secure one time signature scheme* $\mathsf{OTS} = (\mathsf{OTS.Setup}, \mathsf{OTS.Sign}, \mathsf{OTS.Verify})$*, the scheme* $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *described below is a selective CCA secure encryption scheme.*

According to our notation, scheme PKE is a Special-CCA secure encryption scheme.

Notation: Let $\mathsf{PTBE} = (\mathsf{PTBE.Setup}, \mathsf{PTBE.Enc}, \mathsf{PTBE.Dec}, \mathsf{PTBE.Setup\text{-}Alt}, \mathsf{PTBE.Setup\text{-}Alt\text{-}1}, \mathsf{PTBE.Dec\text{-}Alt})$ be a puncturable tag based encryption scheme with message space $\mathcal{X}_n$, tag space $\mathcal{T}_n$ that outputs ciphertexts of size $l(n)$. Let $\mathsf{OTS} = (\mathsf{OTS.Setup}, \mathsf{OTS.Sign}, \mathsf{OTS.Verify})$ be a one time signature scheme that signs messages of length $l(n)$ and the space of verification keys is $\mathcal{T}_n$. Our new scheme PKE has message space $\mathcal{X}_n$.

We now describe the template for building Special-CCA secure encryption schemes from any puncturable tag based encryption. This template can be instantiated by several existing CCA secure encryption schemes in the literature [CHK04,Kil06,PW08].

Construction:

- $\mathsf{PKE.Setup}(1^n)$:
  1. Generate the public key and secret key as $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{PTBE.Setup}(1^n)$.
- $\mathsf{PKE.Enc}(\mathsf{PK}, \mathsf{m})$:
  1. Generate $(\mathsf{VK}, \mathsf{SigK}) \leftarrow \mathsf{OTS.Setup}(1^n)$.
  2. Compute $\mathsf{CT}_1 = \mathsf{PTBE.Enc}(\mathsf{PK}, \mathsf{VK}, \mathsf{m})$ and $\sigma = \mathsf{OTS.Sign}(\mathsf{CT}_1, \mathsf{SigK})$.
  3. Output $\mathsf{CT} = (\mathsf{VK}, \mathsf{CT}_1, \sigma)$ as the ciphertext.
- $\mathsf{PKE.Dec}(\mathsf{SK}, \mathsf{CT})$:
  1. Parse $\mathsf{CT} = (\mathsf{VK}, \mathsf{CT}_1, \sigma)$.
  2. Output $\perp$ if $\mathsf{OTS.Verify}(\mathsf{VK}, \mathsf{CT}_1, \sigma) = 0$.
  3. Output $\mathsf{m} = \mathsf{PTBE.Dec}(\mathsf{SK}, \mathsf{VK}, \mathsf{CT}_1)$.

We prove that the above scheme is CCA-secure in the full version of the paper.

5.3 Instantiating Special-CCA encryption

We show that several popular and well-studied CCA-secure encryption schemes are in fact Special-CCA. That is, they satisfy this property that they can be constructed using PTBE and one-time signatures as shown in the above construction. We now list the encryption schemes below and prove in the full version of the paper that they satisfy the necessary conditions. Formally,

Theorem 4. *The selective CCA-secure encryption schemes in the following popular works are in fact Special-CCA secure encryption schemes:*

– [*CHK04*] *when instantiated with the IBE scheme of* [*BB04*].
– [*CHK04*] *when instantiated with any Hierarchical IBE scheme.*
– [*PW08*] *when instantiated with any Lossy Trapdoor Function.*

5.4 Building selectively secure FE

In this section, we show that the "Special-CCA" secure encryption scheme $\mathsf{PKE} = (\mathsf{PKE.Setup},$
$\mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ from the previous section is FE-Compatible. We prove the security of our construction in two different ways - the first is based on the assumption of sub-exponentially secure indistinguishability obfuscation. Additionally, it requires the one time signature scheme used in the construction of PKE to be a sub-exponentially secure unique signature scheme. On the other hand, the second proof is based on the existence of polynomially secure differing inputs obfuscation and just polynomially secure one time signatures.

Formally, we prove the following two theorems:

Theorem 5. *Any "Special-CCA" secure encryption scheme is selective FE-Compatible for any function family $\mathcal{F}_n$ and poly(n) function key queries assuming:*

– *Sub-exponentially secure indistinguishability obfuscation. (AND)*
– *Sub-exponentially secure unique one time signatures.*

Moreover, the resulting FE scheme is also compact.

Theorem 6. *Any "Special-CCA" secure encryption scheme is selective FE-Compatible for any function family $\mathcal{F}_n$ and poly(n) function key queries assuming:*

– *Polynomially secure differing inputs obfuscation. (AND)*
– *Polynomially secure strong one time signatures.*

Moreover, the resulting FE scheme is also compact.

One example of a one time signature scheme is the Lamport signature scheme[Lam79]. Observe that it is in fact a unique one time signature scheme if we rely on injective one way functions. Instantiating the Special-CCA scheme with the various schemes in Section 5.3, we get the following two corollaries:

Corollary 7 *Let* X *denote the CCA secure encryption scheme in any of the following popular works :*

- *[CHK04] when instantiated with the IBE scheme of [BB04].*
- *[CHK04] when instantiated with any Hierarchical IBE scheme.*
- *[PW08] when instantiated with any Lossy Trapdoor Function.*

Assuming the existence of sub-exponentially secure indistinguishability obfuscation and sub-exponentially secure injective one way functions, scheme X *is selective FE-Compatible for any function family* $\mathcal{F}_n$ *and poly*(n) *function key queries. Moreover, the resulting FE scheme is also compact.*

Corollary 8 *Let* X *denote the CCA secure encryption scheme in any of the following popular works :*

- *[CHK04] when instantiated with the IBE scheme of [BB04].*
- *[CHK04] when instantiated with any Hierarchical IBE scheme.*
- *[PW08] when instantiated with any Lossy Trapdoor Function.*

Assuming polynomially secure differing inputs obfuscation and polynomially secure one way functions, scheme X *is selective FE-Compatible for any function family* $\mathcal{F}_n$ *and poly*(n) *function key queries. Moreover, the resulting FE scheme is also compact.*

Construction: Let $(\mathcal{O}, \mathsf{Eval})$ be a secure obfuscator (note that we will use indistinguishability obfuscation in one proof and differing inputs obfuscation in the other). The functional encryption $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.Keygen}, \mathsf{FE.Dec})$ built from the Special-CCA scheme PKE is as follows. Recall that from the definition of FE-Compatibility, $\mathsf{FE.Setup}(\cdot) = \mathsf{PKE.Setup}(\cdot)$ and $\mathsf{FE.Enc}(\cdot) = \mathsf{PKE.Enc}(\cdot)$.

- $\mathsf{FE.Setup}(1^n)$: Run $\mathsf{PKE.Setup}(1^n)$ to generate $(\mathsf{PK}, \mathsf{SK})$.
- $\mathsf{FE.Enc}(\mathsf{PK}, \mathsf{m})$: Run $\mathsf{PKE.Enc}(\mathsf{PK}, \mathsf{m})$ to generate the ciphertext $\mathsf{CT} = (\mathsf{VK}, \mathsf{CT}_1, \sigma)$.
- $\mathsf{FE.Keygen}(\mathsf{SK}, f)$: Output $\mathsf{SK}_f = \mathcal{O}(\mathsf{G}_f)$ where the program G_f is described below.
- $\mathsf{FE.Dec}(\mathsf{SK}_f, \mathsf{CT})$ Run the program SK_f on input CT to output a string y.

Program G_f

Input : ciphertext CT
Constants : SK

1. Compute $\mathsf{m} = \mathsf{PKE.Dec}(\mathsf{SK}, \mathsf{CT})$.
2. Output $\perp$ if the decryption aborts.
3. Else, output $f(m)$.

Fig. 3: Program for generating function secret key

Security Proof We will prove this via a series of hybrid experiments where we show that every successive pair of hybrids is computationally indistinguishable and the final hybrid is independent of the challenge bit b and hence the attacker's advantage will be 0 in the final hybrid. We will show the indistinguishability of the hybrids using two different proofs in some cases to prove both Theorem 5 and Theorem 6.

- Hyb_1**:** This is the real world experiment with challenge bit b chosen randomly. The challenge ciphertext as $\mathsf{CT}^* = (\mathsf{VK}^*, \mathsf{CT}_1^*, \sigma^*)$.
- Hyb_2**:** This hybrid is identical to the previous hybrid except that now, $\mathsf{FE.Setup}(1^n)$ and the challenge ciphertext are computed differently. Instead of running the setup algorithm $\mathsf{PTBE.Setup}(1^n)$, we now run $\mathsf{PTBE.Setup\text{-}Alt}$ $(1^n, \mathsf{VK}^*, \mathsf{m}_b^*)$ to generate $(\mathsf{PK}, \mathsf{SK}, \mathsf{SK\text{-}Alt}, \mathsf{CT}_1^*)$. The FE scheme's public key is PK, secret key is SK. Now, the challenge ciphertext is computed as follows: generate $(\mathsf{SigK}^*, \mathsf{VK}^*) \leftarrow \mathsf{OTS.Setup}(1^n)$ and compute $\sigma^* = \mathsf{OTS.Sign}(\mathsf{SigK}^*, \mathsf{CT}_1^*)$. The challenge ciphertext is $(\mathsf{VK}^*, \mathsf{CT}_1^*, \sigma^*)$. Note that the alternate secret key $\mathsf{SK\text{-}Alt}$ is not used at all.
- **For each** i **in** $\{0, 1, \ldots, q\}$**,** $\mathsf{Hyb}_{3,i}$**:** This hybrid is identical to the previous hybrid except that now, the function secret key SK_f for the i^{th} function key query f is computed as $\mathcal{O}(\mathsf{G}_f^1)$ for the following program G_f^1.

```
Program $\mathsf{G}_f^1$

Input : ciphertext $\mathsf{CT}$
Constants : $\mathsf{SK}, \mathsf{CT}^* = (\mathsf{VK}^*, \mathsf{CT}_1^*, \sigma^*)$
  1. If $\mathsf{CT} = \mathsf{CT}^*$, output $y$ where $y = f(\mathsf{m}_0^*) = f(\mathsf{m}_1^*)$.
  2. Compute $\mathsf{m} = \mathsf{PKE.Dec}(\mathsf{SK}, \mathsf{CT})$.
  3. Output $\perp$ if the decryption aborts.
  4. Else, output $f(m)$.
```

Fig. 4: Program for generating function secret key

Note that $\mathsf{Hyb}_{3,0}$ corresponds to Hyb_2.

- **For each** i **in** $\{0, 1, \ldots, q\}$**,** $\mathsf{Hyb}_{4,i}$**:** This hybrid is identical to the previous hybrid except that now, the function secret key SK_f for the i^{th} function key query f is computed as $\mathcal{O}(\mathsf{G}_f^2)$ for the following program G_f^2.

```
Program $\mathsf{G}_f^2$

Input : ciphertext $\mathsf{CT} = (\mathsf{VK}, \mathsf{CT}_1, \sigma)$
Constants : $\mathsf{SK}, \mathsf{CT}^* = (\mathsf{VK}^*, \mathsf{CT}_1^*, \sigma^*)$
 1. If $\mathsf{CT} = \mathsf{CT}^*$, output $y$ where $y = f(\mathsf{m}_0^*) = f(\mathsf{m}_1^*)$.
 2. If $\mathsf{VK} = \mathsf{VK}^*$, output $\perp$.
 3. Compute $\mathsf{m} = \mathsf{PKE.Dec}(\mathsf{SK}, \mathsf{CT})$.
 4. Output $\perp$ if the decryption aborts.
 5. Else, output $f(m)$.
```

Fig. 5: Program for generating function secret key

Note that $\mathsf{Hyb}_{4,0}$ corresponds to $\mathsf{Hyb}_{3,q}$.

– **For each** i **in** $\{0, 1, \ldots, q\}$**,** $\mathsf{Hyb}_{5,i}$**:** This hybrid is identical to the previous hybrid except that now, the function secret key SK_f for the i^{th} function key query f is computed as $\mathcal{O}(\mathsf{G}_f^3)$ for the following program G_f^3.

```
Program $\mathsf{G}_f^3$

Input : ciphertext $\mathsf{CT} = (\mathsf{VK}, \mathsf{CT}_1, \sigma)$
Constants : $\mathsf{SK}, \mathsf{CT}^* = (\mathsf{VK}^*, \mathsf{CT}_1^*, \sigma^*)$
 1. If $\mathsf{CT} = \mathsf{CT}^*$, output $y$ where $y = f(\mathsf{m}_0^*) = f(\mathsf{m}_1^*)$.
 2. If $\mathsf{VK} = \mathsf{VK}^*$, output $\perp$.
 3. Check if $\mathsf{OTS.Verify}(\mathsf{VK}, \mathsf{CT}_1, \sigma) = 1$.
 4. Compute $\mathsf{m} = \mathsf{PTBE.Dec\text{-}Alt}(\mathsf{SK\text{-}Alt}, \mathsf{VK}, \mathsf{CT}_1)$.
 5. Output $\perp$ if the decryption aborts or if the signature doesn't verify.
 6. Else, output $f(m)$.
```

Fig. 6: Program for generating function secret key

Note that $\mathsf{Hyb}_{5,0}$ corresponds to $\mathsf{Hyb}_{4,q}$.

– Hyb_6**:** This hybrid is identical to the previous hybrid except that we now run $\mathsf{PTBE.Setup\text{-}Alt\text{-}1}$ $(1^n, \mathsf{VK}^*, \mathsf{m}_\mathsf{b}^*)$ to generate $(\mathsf{PK}, \mathsf{SK}, \mathsf{SK\text{-}Alt}, \mathsf{CT}^*)$.
– Hyb_7**:** This hybrid is identical to the previous hybrid except that we now run $\mathsf{PTBE.Setup\text{-}Alt\text{-}1}(1^n, \mathsf{VK}^*, \mathsf{m}_0^*)$ to generate $(\mathsf{PK}, \mathsf{SK}, \mathsf{SK\text{-}Alt}, \mathsf{CT}^*)$

Observe that in this last hybrid, the challenge ciphertext is created independent of the bit b. Hence, the attacker's advantage in this hybrid is 0.

We refer the reader to the full version for the indistinguishability of every successive pair of hybrids.

References

AJ15. Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 308–326. Springer, 2015.

BB04. Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *Eurocrypt*, 2004.

BCP14. Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 52–73, 2014.

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

BGJS15. Saikrishna Badrinarayanan, Divya Gupta, Abhishek Jain, and Amit Sahai. Multi-input functional encryption for unbounded arity functions. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 27–51. Springer, 2015.

BR94. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, pages 92–111, 1994.

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.

CHK04. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Eurocrypt*, 2004.

FO99. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO*, 1999.

GGG$^+$14. Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602. Springer, 2014.

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

GKW17. Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. *IACR Cryptology ePrint Archive*, 2017.

Had00. Satoshi Hada. Zero-knowledge and code obfuscation. In *ASIACRYPT*, 2000.

HKW15. Susan Hohenberger, Venkata Koppula, and Brent Waters. Universal signature aggregators. In *EUROCRYPT*, 2015.

HLW12. Susan Hohenberger, Allison B. Lewko, and Brent Waters. Detecting dangerous queries: A new approach for chosen ciphertext security. In *EUROCRYPT*, 2012.

Kil06. Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In *TCC*, 2006.

Lam79. Lamport. Constructing digital signatures from a one-way function. *Technical Report SRI-CSL-98, SRI International Computer Science Laboratory*, 1979.

MH14. Takahiro Matsuda and Goichiro Hanaoka. Chosen ciphertext security via UCE. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, volume 8383 of *Lecture Notes in Computer Science*, pages 56–76. Springer, 2014.

MS09. Steven Myers and Abhi Shelat. Bit encryption is complete. In *FOCS*, 2009.

NY90. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, 1990.

PW08. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC*, 2008.

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, 1999.

SW05. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005.

SW08. Amit Sahai and Brent Waters. Slides on functional encryption, powerpoint presentation. 2008.

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

Wat14. Brent Waters. A punctured programming approach to adaptively secure functional encryption. *IACR Cryptology ePrint Archive*, 2014:588, 2014.

WZ17. Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. *IACR Cryptology ePrint Archive*, 2017.