The Trojan Method in Functional Encryption: 
From Selective to Adaptive Security, Generically

Prabhanjan Ananth∗ Zvika Brakerski† Gil Segev‡ Vinod Vaikuntanathan§
UCLA Weizmann Hebrew U MIT

Abstract

In a functional encryption (FE) scheme, the owner of the secret key can generate restricted
decryption keys that allow users to learn specific functions of the encrypted messages and nothing
else. In many known constructions of FE schemes, such a notion of security is guaranteed only
for messages that are fixed ahead of time (i.e., before the adversary even interacts with the
system). This is called selective security, which is too restrictive for many realistic applications.
Achieving adaptive security (also called full security), where security is guaranteed even for
messages that are adaptively chosen at any point in time, seems significantly more challenging.
The handful of known fully-secure schemes are based on specifically tailored techniques that
rely on strong assumptions (such as obfuscation assumptions or multilinear maps assumptions).

In this paper we show that any sufficiently expressive selectively-secure FE scheme can be
transformed into a fully secure one without introducing any additional assumptions. We present
a direct black-box transformation, making novel use of hybrid encryption, a classical technique
that was originally introduced for improving the efficiency of encryption schemes, combined with
a new technique we call the Trojan Method. This method allows to embed a secret execution
thread in the functional keys of the underlying scheme, which will only be activated within the
proof of security of the resulting scheme. As another application of the Trojan Method, we
show how to construct functional encryption schemes for arbitrary circuits starting from ones
for shallow circuits (NC1 or even TC0).

∗E-mail: prabhanjan@cs.ucla.edu. This work was done while visiting MIT, and was supported in part by the
Northrop Grumman Cybersecurity Consortium.
†E-mail: zvika.brakerski@weizmann.ac.il. Supported by an Alon Fellowship and by ISF.
‡E-mail: segev@cs.huji.ac.il. Supported by the European Union’s Seventh Framework Programme (FP7) via
a Marie Curie Career Integration Grant, by the Israel Science Foundation (Grant No. 483/13), and by the Israeli
Centers of Research Excellence (I-CORE) Program (Center No. 4/11).
§Email: vinodv@mit.edu. Research supported in part by DARPA Grant number FA8750-11-2-0225, an Alfred P.
Sloan Research Fellowship, the Northrop Grumman Cybersecurity Research Consortium (CRC), the Qatar Computing
Research Institute, Microsoft Faculty Fellowship, and a Steven and Renee Finn Career Development Chair from MIT.
1 Introduction

Traditional notions of public-key encryption provide all-or-nothing access to data: owners of the secret key can recover the entire message from a ciphertext, whereas those who do not know the secret key learn nothing at all. Functional encryption [SW05, KSW08, LOS+10, BSW11, O’N10] is a modern type of encryption scheme where the owner of the (master) secret key can release function-specific secret keys $sk_f$, which enable a user to compute $f(x)$ from an encryption of $x$. Furthermore, using the key $sk_f$, one should not be able to extract any information about $x$ other than $f(x)$. There are a number of ways to formalize this abstract requirement. In this paper we will exclusively use an indistinguishability based definition, requiring that given $x_0, x_1$ such that $f(x_0) = f(x_1)$, it is computationally hard to distinguish between $(\text{Enc}(x_0), sk_f)$ and $(\text{Enc}(x_1), sk_f)$. This definition extends to multiple messages and keys in a fairly straightforward manner.

While initial constructions of functional encryption [BF03, BCOP04, KSW08, LOS+10] were limited to simple function classes such as point functions and inner products, recent developments have dramatically improved the state of the art in this area. In particular, the works of Sahai and Seyalioglu [SS10] and Gorbunov, Vaikuntanathan and Wee [GVW12] showed that if only a single function key is produced, then functional encryption can be based on any semantically secure encryption. This result can be extended to a case where the number of function keys is polynomial and known a-priori. Goldwasser, Kalai, Popa, Vaikuntanathan and Zeldovich [GKP+13] constructed a scheme with succinct ciphertexts based on a specific hardness assumption (Learning with Errors).

The first functional encryption scheme that supports a-priori unbounded number of function keys was recently introduced by Garg, Gentry, Halevi, Raykova, Sahai and Waters [GGH+13], based on the existence of secure indistinguishability obfuscation (to which a heuristic construction is presented in the same paper). Garg et al. showed that given a secure indistinguishability obfuscator, their functional encryption scheme is selectively secure.

At a high level, selective security guarantees that the scheme is secure so long as the messages being encrypted are chosen independently of the parameters of the scheme. This is reflected in the formal definition by requiring the adversary to specify the messages that he intends to encrypt before seeing any parameter of the scheme, in particular before seeing any functional key. If the message to be encrypted depends on the the parameters (e.g. on the public key or on function keys) then security cannot be guaranteed. Whereas the independence assumption is justified in some cases, the ultimate sought-after notion of security is adaptive security (often called full security). An adaptively secure scheme guarantees security regardless of how the encrypted messages are chosen (granted that they can be produced in polynomial time, of course). This is reflected in the formal definition by allowing the adversary to specify the messages $x_0, x_1$ at any point in its execution, even after seeing function keys and encryptions of other ciphertexts.

Historically, the first functional encryption schemes have been selectively secure [BB04, GPSW06, KSW08, GVW13, GKP+13]. The question of constructing adaptively secure functional encryption is considered to be notoriously difficult and only few approaches are known. The basic observation is that if the input space is bounded in size, e.g. $\{0,1\}^n$ for a known $n$, then the $x$ values can be guessed ahead of time with probability $2^{-n}$. Starting with a sub-exponential hardness assumption, and taking the security parameter to be polynomial in $n$, allows to argue that the selectively secure scheme is in fact also adaptively secure. This method is known as “complexity leveraging” and is not considered to be a satisfactory solution to the problem since it requires relying on strong hardness assumptions. The powerful “dual system encryption” method, introduced by Waters [Wat09], had
been used to construct adaptively secure attribute based encryption (a weak version of functional encryption) for formulas, as well as an adaptively secure functional encryption scheme for linear functions [LOS+10]. However, this method is merely a general outline and each construction is required to tailor the solution based on its specialized assumption. In some cases, such as attribute based encryption for circuits, it is not known how to implement dual system encryption or any other method to achieve adaptive security.

Starting with [GGH+13], there has been effort in the research community to construct an adaptively secure functional encryption scheme for polynomial size circuits with a-priori unbounded many function keys. Boyle, Chung and Pass [BCP14] constructed an adaptively secure FE scheme, under the assumption that extractability obfuscators exist (these are stronger primitives than the indistinguishability obfuscators used by [GGH+13]). Waters [Wat14] showed how to do this assuming indistinguishability obfuscation, thus matching the assumption of [GGH+13]. Garg, Gentry, Halevi and Zhandry [GGHZ14] showed how to construct adaptively secure public-key FE from non-standard assumptions on multilinear maps. Each of these constructions uses its own methods and techniques, and in general it was not known how to achieve adaptivity in any non-ad-hoc method.

In this work, we use new techniques to show a generic construction of adaptively secure functional encryption, starting from any selectively secure scheme.

1.1 Our Results: From Selective to Adaptive Security

We show that any selectively secure functional encryption scheme implies an adaptively secure scheme, without relying on any additional assumptions. Our transformation applies equally to symmetric-key or public-key functional encryption, where the resulting adaptive scheme inherits its symmetric-key or public-key properties from the building block scheme. The following theorem summarizes our main contribution.

**Theorem 1 (informal).** Given any public-key (respectively symmetric-key) selectively-secure functional encryption scheme, there exists an adaptively secure public-key (respectively symmetric-key) functional encryption scheme supporting the class of bounded polynomial size circuits.

We require that the selective scheme supports a sufficiently rich function class. In particular it will be applied to functions with a-priori bounded polynomial size and a-priori bounded depth (essentially the depth of computing a weak pseudorandom function, which is logarithmic under mild assumptions). The resulting adaptively secure scheme, however, does not have a depth bound. Therefore a corollary of our construction is that selectively secure functional encryption with depth bound implies adaptively secure functional encryption without a depth bound. When combined with the result by [GGH+13], we get an adaptively secure functional encryption scheme based on indistinguishability obfuscation (and one-way functions). This can be viewed as a simpler alternative to Waters [Wat14] who achieves the same result.

We view the significance of our result in a number of dimensions. First of all, it answers the basic call of cryptographic research to substantiate the existence of complicated primitives on that of simple primitives. We feel that this is of special interest in the case of adaptive security where it seemed that ad-hoc methods were required. Secondly, our construction, being of fairly low overhead, will allow to focus the attention of the research community in studying selectively secure functional encryption, rather than investing unwarranted efforts in the study of adaptive security. Lastly, we hope that our methods can be extended and employed towards those variants for which adaptive security is yet unattained, such as attribute based encryption for all polynomial size circuits.
1.2 Our Techniques: The Trojan Method and Hybrid Encryption

Our result is achieved by incorporating a number of techniques which will be explained in this section. We start by presenting a technique that we call The Trojan Method, whose origins lie in the “trapdoor circuits” idea presented in the work of De Caro et al. [CIJ+13], and in the work of Brakerski and Segev [BS14] in the context of function-private private-key FE. We extend this method and show that it is in fact a very powerful tool in the context of public-key FE, and as an example show that FE for “shallow” circuits (a circuit family that allows to compute weak PRFs) can be extended to one that applies to all circuits. We then explain the power of hybrid encryption in the context of functional encryption, and put it together with the Trojan method to present our final construction. Finally, a short comparison with of our technique with the aforementioned “dual system encryption” technique that had been used to achieve adaptively secure attribute based encryption.

The Trojan Method. The Trojan Method is a way to embed a hidden functionality thread in an FE secret-key that can only be invoked by special ciphertexts generated using special (secret) back-door information. This thread remains completely unused in the normal operation of the scheme (and can be instantiated with meaningless functionality). In the proof, however, the secret thread will be activated by the challenge ciphertext in such a way that is indistinguishable to the user (= attacker). Namely, the user will not be able to tell that it is executing the secret thread and not the main thread. This will be extremely beneficial to prove security. We wish to argue that in the view of the user, the execution of the main thread does not allow to distinguish between the encryption of two messages \(x_0, x_1\). The problem is that for functionality purposes, the main thread has to know which input it is working on. This is where the hidden thread comes into the play. We will design the hidden thread so that in the eyes of the user, it is computationally indistinguishable from the main thread on the special messages \(x_0, x_1\). However, in the hidden thread, the output can be computed in a way that does not distinguish between \(x_0\) and \(x_1\) (either by a statistical or a computational argument), which will allow us to conclude that encryptions of \(x_0, x_1\) are indistinguishable.

Technically, the hidden thread is implemented using (standard) symmetric-key encryption, which in turn can be constructed starting with any one-way function. In the functional secret-key generation process for a function \(f\), the secret-key generation process will produce a symmetric-key ciphertext \(c\) (which can just be encryption of 0 or another fixed message, since it only needs to have meaningful content in the security proof). It will then consider the function \(G_{f,c}\) that takes as input a pair \((x, s)\), and first checks whether it can decrypt \(c\) using \(s\) as a symmetric key. If it cannot, then it just runs \(f\) on \(x\) and returns the output. If \(s\) actually decrypts \(c\), we consider \(f^* = \text{Dec}_s(c)\), and the output is the execution of \(f^*\) on \(x\). The value \(c\) is therefore used as a Trojan Horse: Its contents are hidden from the users of the scheme, however given a hidden command (in the form of the symmetric \(s\)) it can embed functionality that “takes over” the functional secret-key.

We note that in order to support the Trojan method, our FE scheme must support a rich enough class of circuits which allows branch operations, symmetric decryption and execution of the decrypted \(f^*\).

This method can be seen as a weak form of function privacy in FE, but one that can be applied even in the context of public-key FE. In essence, we cannot hide the main thread of the evaluated function (this is unavoidable in public-key FE). However, we can hide the secret thread and thus allow the function to operate in a designated way for specially generated ciphertexts.
For a fairly simple example on how the Trojan method is applied, see the outline for our shallow-to-deep transformation below.

**Example: Shallow-to-Deep FE.** We use the Trojan method to show that FE that only supports secret-keys for functions with shallow circuits (e.g. logarithmic depth) implies a scheme that works for circuits of arbitrary depth (although with a size bound).

The idea is fairly natural: Instead of producing a secret-key for the function, we produce a key for a shallow randomized encoding (RE) of the function (in particular, we use garbled circuits). A randomized encoding [IK00] is a randomized process such that given \(f, x\), \(\text{RE}(f, x) = \text{RE}(f, x; \text{rand})\) is a randomized function that can be computed by a shallow circuit (assuming weak PRFs can be computed by shallow circuits). Furthermore, there is an efficient way to retrieve \(f(x)\) from \(\text{RE}(f, x)\), however nothing except \(f(x)\) is revealed. The latter is guaranteed by the existence of a simulator such that \(\text{Sim}(f(x)) \cong \text{RE}(f, x)\). We note that a similar approach had been attempted to leverage the supported depth of obfuscators [App13], however it appears to only work for strong notions of obfuscation such as virtual-black-box that are impossible to achieve in general.

Using the Trojan method, we show how this can be achieved for FE. As mentioned above, whenever the scheme asks for a secret-key for \(f\), it will in fact receive a secret-key for a function \(G\) that given \(x\), outputs \(G_{f,t}(x, k) = \text{RE}(f, x; \text{PRF}_k(t))\). To encrypt a value \(x\), we choose \(k\) at random and encrypt the pair \((x, k)\). The decryption process will include applying the functional secret key to obtain \(\text{RE}(f, x)\) and computing \(f(x)\) from that value. For security we have to show that if \(x_0, x_1\) are such that \(f(x_0) = f(x_1)\) then their encryptions are indistinguishable. This is where the Trojan comes into the play. Consider, for starters, the case where \(x_0\) had been encrypted (with some PRF seed \(k^*\)). We will insert a Trojan thread into \(G_{f,t}\), that, once activated, simply outputs \(\text{RE}(f, x_0; \text{PRF}_k(t))\). Now, the hidden thread outputs exactly the same as the main thread, and therefore switching to the hidden thread will go unnoticed by the user (note that the hidden thread is only activated for the special encryption of \((x_0, k^*)\)). Once the switch had been made, \(x_0, k^*\) no longer need to appear in the ciphertext (since all the information about them had been embedded in the hidden thread). The hidden thread is indistinguishable from just outputting \(\text{RE}(f, x_0; \text{rand})\), which is in turn indistinguishable from \(\text{Sim}(f(x_0))\), which is in turn identical to \(\text{Sim}(f(x_1))\). It follows that the user cannot distinguish between the case where \(x_0\) had been encrypted and the case where \(x_1\) had been encrypted.

**Hybrid Functional Encryption.** The second technique, in addition to the Trojan method, that we employ to achieve our main result is hybrid encryption. Hybrid encryption is a veteran technique in cryptography which has been used in a variety of settings. We show that in the context of functional encryption it is especially powerful.

The idea in functional encryption is to combine two encryption schemes: An “external” scheme (sometimes called KEM – Key Encapsulation Mechanism) one and in “internal” scheme (sometimes called DEM – Data Encapsulation Mechanism). In order to encrypt a message in the hybrid scheme, a fresh key is generated for the internal scheme, and is used to encrypt the message. Then the key itself is encrypted using the external scheme. The final hybrid ciphertext contains the two ciphertexts: \((\text{Enc}_{\text{ext}}(k), \text{Enc}_{\text{int}, k}(m))\) (all external ciphertexts use the same key). To decrypt, one first decrypts the external ciphertext, retrieves \(k\) and applies it to the internal ciphertext. Note that if, for example, the external scheme is public-key and the internal is symmetric key, then the resulting scheme will also be public key. Hybrid encryption is often used in cases where the external
scheme is less efficient (e.g. in encrypting long messages) and thus there is an advantage in using it
to encrypt only a short key, and encrypt the long message using the more efficient internal scheme.
Lastly, note that the internal scheme only needs to be able to securely encrypt a single message.

The intuition as to why hybrid encryption may be good for achieving adaptive security is that
the external scheme only encrypts keys for the internal scheme. Namely, it only encrypts messages
from a predetermined and known distribution, so selectively secure scheme should be enough for
the external scheme. The hardness of adaptive security is "pushed" to the internal scheme, but
there the task is easier since the internal scheme only needs to be able to encrypt a single message,
and it can be private-key rather than public-key.

Let us see how to employ this idea in the case where both internal and external schemes are
FE schemes. To encrypt, we will generate a fresh master secret key for the internal scheme, and
encrypt it under the external scheme. To generate a key for the function \( f \), the idea is to generate
a key for the function \( G_f(\text{msk}_{\text{int}}) \) which takes a master key for the internal scheme, and outputs a
secret key for function \( f \) under the internal scheme, using \( \text{msk}_{\text{int}} \) (randomness is handled using a
PRF as in the shallow-to-deep example). This will allow to decrypt in a two-step process as above.
First apply the external secret-key for \( G_f \) to the external ciphertext, this will give you an internal
secret key for \( f \), which is in turn applied to the internal ciphertext to produce \( f(x) \).

For the external scheme, we will use a selectively secure FE scheme (for the sake of concreteness,
let us say public-key FE). As explained above, selective security is sufficient here since all the
messages encrypted using the external scheme can be generated ahead of time (i.e. they do not
depend on the actual \( x \)'s that the user wishes to encrypt).

For the internal scheme, we require an FE scheme that is adaptively secure, but only sup-
ports the encryption of a single message. Fortunately, such a primitive can be derived from
the works of [GVW12, BS14]. In [GVW12], the authors present an adaptively secure one-time
bounded FE scheme. This scheme allows to only generate a key for one function, and to encrypt
as many messages as the user wishes. This construction is based on the existence of semantically
secure encryption, so the public-key version needs public-key encryption and the symmetric version
needs symmetric encryption. While this primitive seems dual to what we need for our purposes,
[BS14] shows how to transform private-key FE schemes into function private FE. In function-
private FE, messages and functions enjoy the same level of privacy, in the sense that a user that
produces \( x_0, x_1, f_0, f_1 \) such that \( f_0(x_0) = f_1(x_1) \) cannot distinguish between \( (\text{Enc}(x_0), \text{sk}_{f_0}) \) and
(\( \text{Enc}(x_1), \text{sk}_{f_1} \)). Therefore, after applying the [BS14] transformation, we can switch the roles of the
functions and messages, and obtain a symmetric FE scheme which is adaptively secure for a single
message and many functions. (We note that the symmetric version of the [GVW12] scheme can be
shown to be function private even without the [BS14] transformation, however since this claim is
not made explicitly in the paper we choose not to rely on it.)

Putting the two components together produces our final scheme. In the proof, we will use the
Trojan method to embed a hidden thread in which \( \text{msk}_{\text{int}} \) is not used at all, but rather \( G_f \) produces
a precomputed internal \( \text{sk}_f \). This will allow us to remove \( \text{msk}_{\text{int}} \) from the challenge ciphertext
and use the security properties of the internal scheme to argue that a internal encryption of \( x_0, x_1 \) are
identical so long as \( f(x_0) = f(x_1) \).

Relation to Dual-System Encryption. Our approach takes some resemblance to the “Dual-
System Encryption” method of Waters [Wat09] and followup works [LW10, LW12]. This method
had been used to prove adaptive security for Identity Based Encryption and Attribute Based
Encryption, based on the hardness of some problems on groups with bilinear-maps. In broad terms, in their proof the distribution of the ciphertext is changed into “semi-functional” mode in a way that is indiscoverable by an observer. A semi-functional ciphertext is still decryptable by normal secret keys. Then, the secret-keys are modified into semi-functional form, which is useless in decrypting semi-functional ciphertexts. This is useful since in IBE and ABE, the challenge ciphertext is not supposed to be decryptable by those keys given to the adversary. Still, a host of algebraic techniques are used to justify the adversary’s inability to produce other semi-functional ciphertexts in addition to the challenge, which would foil the reduction.

Our proof technique also requires changing the distributions of the keys and challenge ciphertext. However, there are also major differences. Our modified ciphertext is not allowed to interact with properly generated secret keys, and therefore the distinction between “normal” and “semi-functional” does not fit here. Furthermore, in Identity Based and Attribute Based Encryption, the attacker in the security game is not allowed to receive keys that reveal any information on the message, whereas in our case, there is a structured and well-defined output for any ciphertext and any key. This means that the information required for decryption (which can be a-priori unbounded) needs to be embedded in the keys. Lastly, our proof is completely generic and does not rely on the algebraic structure of the underlying hardness assumption as in previous implementations of this method.

2 Preliminaries

We let \( \lambda \) denote the security parameter. We say that a function \( \mu(\lambda) \) is negligible if for any polynomial \( p(\lambda) \) it holds that \( \mu(\lambda) < 1/p(\lambda) \) for all sufficiently large \( \lambda \in \mathbb{N} \).

2.1 Public-key Functional Encryption

A public-key functional encryption (FE) scheme \( \Pi_{\text{pub}} \), defined for a class of functions \( F = \{F_{\lambda}\}_{\lambda \in \mathbb{N}} \) and message space \( M = \{M_{\lambda}\}_{\lambda \in \mathbb{N}} \), is represented by four PPT algorithms (\( \text{Pub.Setup}, \text{Pub.KeyGen}, \text{Pub.Enc}, \text{Pub.Dec} \)). The input length of any \( f \in F_{\lambda} \) is the same as the length of any \( m \in M_{\lambda} \). The description of these four algorithms is given below.

- \( \text{Pub.Setup}(1^{\lambda}) \): It takes as input a security parameter \( \lambda \) in unary and outputs a public key-secret key pair (MPK, MSK).
- \( \text{Pub.KeyGen}(\text{MSK}, f \in F_{\lambda}) \): It takes as input a secret key MSK, a function \( f \in F_{\lambda} \) and outputs a functional key \( sk_f \).
- \( \text{Pub.Enc}(\text{MPK}, m \in M_{\lambda}) \): It takes as input a public key MPK, a message \( m \in M_{\lambda} \) and outputs an encryption of \( m \).
- \( \text{Pub.Dec}(sk_f, \text{CT}) \): It takes as input a functional key \( sk_f \), a ciphertext CT and outputs \( \hat{m} \).

A public-key FE scheme is defined for a complexity class \( \mathcal{C} \) if the public-key FE scheme is defined for \( \mathcal{F} \), which consists of all the functions that can be implemented by circuits in \( \mathcal{C} \).

The correctness notion of a FE scheme dictates that there exists a negligible function \( \text{negl}(\lambda) \) such that for all sufficiently large \( \lambda \in \mathbb{N} \), for every message \( m \in M_{\lambda} \), and for every function \( f \in F_{\lambda} \),
it holds that $\Pr[f(m) \leftarrow \text{Pub.Dec(Pub.KeyGen(MSK, f), Pub.Enc(MPK, m))}] \geq 1 - \mu(\lambda)$, where $(\text{MPK, MSK}) \leftarrow \text{Pub.Setup}(1^{\lambda})$, and the probability is taken over the random choices of all algorithms.

There are different ways to model the security of a functional encryption scheme and the security notion of interest to this work is the indistinguishability-based notion. At a high level, indistinguishability-based notion of security, modeled as a game between the challenger and a PPT adversary, states that the adversary cannot distinguish (with probability significantly different from 1/2), the encryptions of two messages even after being given functional keys which should correspond to the functions (of adversary’s choice) that evaluate to the same output on both the messages – note that this condition is required to prevent the adversary from trivially distinguishing the two ciphertexts.

In the security game, if the adversary is supposed to declare the challenge messages even before it sees the public parameters from the challenger then the FE scheme that satisfies this notion is said to be selectively-secure. If the adversary can declare the challenge messages at any time during the game then the FE scheme satisfying such a notion is said to be adaptively-secure. Clearly, adaptively-secure FE is at least as strong as selectively-secure FE. We give the formal definitions\(^1\) of a selectively-secure public-key FE as well as an adaptively-secure public-key FE below.

**Definition 1** (Selectively-secure public-key FE). A public-key functional encryption scheme $\text{Sel} = (\text{Sel.Setup}, \text{Sel.KeyGen}, \text{Sel.Enc}, \text{Sel.Dec})$ over a function space $\mathcal{F} = \{F_{\lambda}\}_{\lambda \in \mathbb{N}}$ and a message space $\mathcal{M} = \{M_{\lambda}\}_{\lambda \in \mathbb{N}}$ is a **selectively-secure public-key functional encryption scheme** if for any PPT adversary $A$ there exists a negligible function $\mu(\lambda)$ such that the advantage of $A$ is defined to be

$$\text{Adv}^{\text{Sel}}_{A} = \left| \Pr[\text{Expt}^{\text{Sel}}_{A}(1^{\lambda}, 0) = 1] - \Pr[\text{Expt}^{\text{Sel}}_{A}(1^{\lambda}, 1) = 1] \right| \leq \mu(\lambda),$$

where for each $b \in \{0, 1\}$ and $\lambda \in \mathbb{N}$ the experiment $\text{Expt}^{\text{Sel}}_{A}(1^{\lambda}, b)$, modeled as a game between the challenger and the adversary $A$, is defined as follows:

1. The adversary submits the challenge message-pair $(m_0, m_1)$ to the challenger.

2. The challenger executes $\text{Sel.Setup}(1^{\lambda})$ to obtain $(\text{Sel.MPK}, \text{Sel.MSK})$. It then executes $\text{Sel.Enc}(\text{Sel.MPK}, m_b)$ to obtain CT. The challenger sends $(\text{Sel.MPK}, \text{CT})$ to the adversary.

3. **Query phase**: For every function query $f$ submitted by the adversary, the challenger generates $\text{Sel.sk}_f$, where $\text{Sel.sk}_f$ is the output of $\text{Sel.KeyGen}(\text{Sel.MSK}, f)$. The challenger sends $\text{Sel.sk}_f$ only if $f(m_0) = f(m_1)$. Otherwise, it aborts.

4. The output of the experiment is $b'$, where $b'$ is the output of $A$.

**Definition 2** (Adaptively-secure public-key FE). A public-key functional encryption scheme $\text{Ad} = (\text{Ad.Setup}, \text{Ad.KeyGen}, \text{Ad.Enc}, \text{Ad.Dec})$ over a function space $\mathcal{F} = \{F_{\lambda}\}_{\lambda \in \mathbb{N}}$ and a message space $\mathcal{M} = \{M_{\lambda}\}_{\lambda \in \mathbb{N}}$ is a **adaptively-secure public-key functional encryption scheme** if for any PPT adversary $A$ there exists a negligible function $\mu(\lambda)$ such that the advantage of $A$ is defined to be

$$\text{Adv}^{\text{Ad}}_{A} = \left| \Pr[\text{Expt}^{\text{Ad}}_{A}(1^{\lambda}, 0) = 1] - \Pr[\text{Expt}^{\text{Ad}}_{A}(1^{\lambda}, 1) = 1] \right| \leq \mu(\lambda),$$

\(^1\)Even though the definitions talk about single message challenge queries, it can be easily extended to the case when the adversary can ask multiple message queries. In the public key setting, both the definitions are equivalent by a standard hybrid argument.
where for each \( b \in \{0, 1\} \) and \( \lambda \in \mathbb{N} \) the experiment \( \text{Expt}^{\text{Ad}}_{\lambda}(1^{\lambda}, b) \), modeled as a game between the challenger and the adversary \( A \), is defined as follows:

1. The challenger first executes \( \text{Ad.Setup}(1^{\lambda}) \) to obtain \((\text{Ad.MPK}, \text{Ad.MSK})\). It then sends \( \text{Ad.MPK} \) to the adversary.

2. **Query Phase I:** The adversary submits a function query \( f \) to the challenger. The challenger sends back \( \text{Ad.sk}_f \) to the adversary, where \( \text{Ad.sk}_f \) is the output of \( \text{Ad.KeyGen}(\text{Ad.MSK}, f) \).

3. **Challenge Phase:** The adversary submits a message-pair \((m_0, m_1)\) to the challenger. The challenger checks whether \( f(m_0) = f(m_1) \) for all function queries \( f \) made so far. If this is not the case, the challenger aborts. Otherwise, the challenger sends back \( CT = \text{Ad.Enc}(\text{Ad.MSK}, m_b) \).

4. **Query Phase II:** The adversary submits a function query \( f \) to the challenger. The challenger generates \( \text{Ad.sk}_f \), where \( \text{Ad.sk}_f \) is the output of \( \text{Ad.KeyGen}(\text{Ad.MSK}, f) \). It sends \( \text{Ad.sk}_f \) to the adversary only if \( f(m_0) = f(m_1) \), otherwise it aborts.

5. The output of the experiment is \( b' \), where \( b' \) is the output of \( A \).

### 3 From Selective FE to Adaptive FE

In this section we present our transformation from selective to adaptive security in the public-key setting. The transformation in the symmetric-key setting is essentially identical and is provided in the full version due to space limitations.

One of our building blocks is a private-key FE scheme which is adaptively secure for a single message query and many ciphertext queries. A bit more formally, a single-ciphertext private-key functional encryption scheme is a functional encryption scheme where the adversary in the security game (either selective or adaptive) is allowed to submit only a single challenge message query. For our selective to adaptive transformation, we require a single-ciphertext private-key FE scheme that satisfies adaptive security. This scheme can be constructed based on any one-way function using the works of [GVW12, BS14] (see Section 1.2 for a sketch of this construction). A formal treatment of this scheme is provided in the full version. Henceforth, for brevity, we refer to adaptively secure single-ciphertext private-key FE scheme just as single-ciphertext FE scheme.

Our transformation relies on the following building blocks:

1. Selectively-secure public-key FE scheme, \( \text{Sel} = (\text{Sel.Setup}, \text{Sel.KeyGen}, \text{Sel.Enc}, \text{Sel.Dec}) \).

2. Single-ciphertext FE scheme, \( \text{OneCT} = (\text{OneCT.Setup}, \text{OneCT.KeyGen}, \text{OneCT.Enc}, \text{OneCT.Dec}) \).

3. Symmetric-key encryption scheme with pseudorandom ciphertexts\(^2\), \( \text{SYM} = (\text{Sym.Setup}, \text{Sym.Enc}, \text{Sym.Dec}) \).

4. Pseudorandom function family denoted by \( \mathcal{F} \) and the space of PRF keys is represented by \( \mathcal{K} \).

Our construction of an adaptively-secure public-key FE, denoted by \( \text{Ad} \), is as follows.

\(^2\)Such a scheme can be constructed based on any PRF [Gol09].
Ad.Setup(1\(^λ\)): Execute Sel.Setup(1\(^λ\)) to obtain (Sel.MPK, Sel.MSK). Output (Ad.MPK = Sel.MPK, Ad.MSK = Sel.MSK).

Ad.KeyGen(Ad.MSK = Sel.MSK, \(f\)):

- Pick a uniformly random string \(C_E \leftarrow \{0,1\}^{\ell_1(\lambda)}\) and a uniformly random tag \(\tau \leftarrow \{0,1\}^{\ell_2(\lambda)}\).
- Define the circuit
  
  \[ G_{f,C_E,\tau} (\text{OneCT.SK}, K, \text{Sym.K}, \beta) = \begin{cases} 
  \text{OneCT.sk}_f & \text{if } \beta = 0 \\
  \text{Sym.Dec}(\text{Sym.K}, C_E) & \text{if } \beta = 1 
  \end{cases} \]

  where \(\text{OneCT.sk}_f\) is the output of \(\text{OneCT.KeyGen}(\text{OneCT.SK}, f; \text{PRF}_K(\tau))\) and \(C_E \in \{0,1\}^{\ell_1(\lambda)}, \tau \in \{0,1\}^{\ell_2(\lambda)}\) are as above. Furthermore, \(K, \text{Sym.K} \in \{0,1\}^\lambda\), and \(\beta \in \{0,1\}\).

- Run \(\text{Sel.sk}_G \leftarrow \text{Sel.KeyGen}(\text{Sel.MSK}, G_{f,C_E,\tau})\) and output \(\text{Ad.sk}_f = \text{Sel.sk}_G\).

Ad.Enc(Ad.MPK = Sel.MPK, \(m\)):

- Execute \(\text{OneCT.Setup}(1^\lambda)\) to obtain \(\text{OneCT.SK}\).
- Sample \(K\) from the appropriate PRF key space.
- Execute \(\text{OneCT.Enc}(\text{OneCT.SK}, m)\) to obtain \(CT_0\). Execute \(\text{Sel.Enc}(\text{Sel.MPK}, M = (\text{OneCT.SK}, K, 0^\lambda, 0))\) to obtain \(CT_1\).
- Output \(CT = (CT_0, CT_1)\).

Ad.Dec(Ad.sk\(_f\) = Sel.sk\(_G\), CT = (CT\(_0\), CT\(_1\))): Execute \(\text{Sel.Dec}(\text{Sel.sk}_G, CT_1)\) to obtain \(\text{OneCT.sk}_f\). Execute \(\text{OneCT.Dec}(\text{OneCT.sk}_f, CT_0)\) to obtain \(\hat{m}\).

The correctness of the above scheme is shown in the full version. We prove the security below.

**Security.** We show that any PPT adversary \(\mathcal{A}\) succeeds in the adaptive security game of Ad with only negligible probability. We will show this in a sequence of hybrids. We denote the advantage of the adversary in \(\text{Hybrid}_{i,b}\) to be the probability that the adversary outputs 1 in this hybrid and this quantity is denoted by \(\text{Adv}^A_{i,b}\). For \(b \in \{0,1\}\), we define the following hybrids.

**Hybrid\(_1,b\):** This corresponds to the real experiment when the challenger encrypts the \(b^{th}\) message in the message pair submitted by the adversary. More precisely, if the adversary submits the message pair \((m_0, m_1)\) to the challenger, the challenger then sends the challenge ciphertext \(CT^*\) back to the adversary, where \(CT^*\) is the encryption of message \(m_b\). The output of this hybrid is the same as the output of the adversary.

**Hybrid\(_2,b\):** The challenger replaces \(C_E\) in every functional key (each key has a different \(C_E\)), corresponding to the query \(f\) made by the adversary, with a symmetric key encryption of \(\text{OneCT.sk}_f\). Here, \(\text{OneCT.sk}_f\) is the output of \(\text{OneCT.KeyGen}(\text{OneCT.SK}^*, f; \text{PRF}_K^*(\tau))\) and \(K^*\) is a PRF key drawn from the keyspace \(K\). Further, the symmetric encryption is computed with respect to \(\text{Sym.K}^*\),
where $\text{Sym}.K^*$ is the output of $\text{Sym}.\text{Setup}(1^\lambda)$ and $\tau$ is the tag associated to the functional key of $f$. We emphasize that the same $\text{Sym}.K^*$ and $K^*$ are used while generating all the functional keys. Further, the challenger generates the challenge ciphertext $\text{CT}^*$ to be $(\text{CT}^*_0, \text{CT}^*_1)$, where $\text{CT}^*_0$ is the output of $\text{OneCT}.\text{Enc}(\text{OneCT}.\text{SK}^*, m_b)$ and $\text{CT}^*_1$ is the output of $\text{Sel}.\text{Enc}(\text{Sel}.\text{MSK}, (\text{OneCT}.\text{SK}^*, K^*, 0^\lambda, 0))$. The rest of the hybrid is the same as the previous hybrid, $\text{Hybrid}_{1,b}$.

$\text{Hybrid}_{1,b}$ and $\text{Hybrid}_{2,b}$ are computationally indistinguishable assuming the pseudorandomness of the ciphertexts produced by $\text{SYM}$.

$\text{Hybrid}_{3,b}$: The challenger modifies the challenge ciphertext $\text{CT}^* = (\text{CT}^*_0, \text{CT}^*_1)$. It generates $\text{CT}^*_1$ to be an encryption of the message $(0^\lambda, 0^\lambda, \text{Sym}.K^*, 1)$. The ciphertext component $\text{CT}^*_0$ is generated the same way as before. In more detail, the challenge ciphertext is now $\text{CT}^* = (\text{CT}^*_0 = \text{OneCT}.\text{Enc}(\text{OneCT}.\text{SK}^*, m_b), \text{CT}^*_1 = \text{Sel}.\text{Enc}(\text{Sel}.\text{MPK}, (0^\lambda, 0^\lambda, \text{Sym}.K^*, 1)))$. The rest of the hybrid is the same as the previous hybrid, $\text{Hybrid}_{2,b}$.

$\text{Hybrid}_{2,b}$ and $\text{Hybrid}_{3,b}$ are computationally indistinguishable assuming the selective security of $\text{Sel}$.

$\text{Hybrid}_{4,b}$: For every functional query $f$ made by the adversary, the challenger generates $C_E$ by executing $\text{Sym}.\text{Enc}(\text{Sym}.K^*, \text{OneCT}.sk_f)$, with $\text{OneCT}.sk_f$ being the output of $\text{OneCT}.\text{KeyGen}(\text{OneCT}.\text{SK}^*, f; R)$, where $R$ is picked at random. The rest of the hybrid is the same as the previous hybrid.

$\text{Hybrid}_{3,b}$ and $\text{Hybrid}_{4,b}$ are computationally indistinguishable assuming the security of pseudorandom function family $\mathcal{F}$. Further, assuming the adaptive security of $\text{OneCT}$, we can show that $\text{Hybrid}_{4,0}$ is computationally indistinguishable from $\text{Hybrid}_{4,1}$.

The above claims implies that $\text{Hybrid}_{1,0}$ is computationally indistinguishable from $\text{Hybrid}_{1,1}$ which proves the adaptive security of $\text{Ad}$. We thus have the following theorem.

**Theorem 2.** There exists an adaptively-secure public-key FE scheme assuming the existence of a sufficiently-expressive selectively-secure public-key FE scheme, adaptively-secure single-ciphertext private-key FE scheme, pseudorandom functions and a symmetric-key encryption scheme with pseudorandom ciphertexts.

As remarked earlier, all of the building blocks that are specified in the above theorem are implies by the existence of a sufficiently-expressive selectively-secure public-key FE scheme

**References**


[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, Advances in Cryptology - CRYPTO 2009,
Abstract

In a functional encryption (FE) scheme, the owner of the secret key can generate restricted decryption keys that allow users to learn specific functions of the encrypted messages and nothing else. In many known constructions of FE schemes, security is guaranteed only for messages that are fixed ahead of time (i.e., before the adversary even interacts with the system). This so-called selective security is too restrictive for many realistic applications. Achieving adaptive security (also called full security), where security is guaranteed even for messages that are adaptively chosen at any point in time, seems significantly more challenging. The handful of known adaptively-secure schemes are based on specifically tailored techniques that rely on strong assumptions (such as obfuscation or multilinear maps assumptions).

We show that any sufficiently-expressive selectively-secure FE scheme can be transformed into an adaptively-secure one without introducing any additional assumptions. We present a black-box transformation, for both public-key and private-key schemes, making novel use of hybrid encryption, a classical technique that was originally introduced for improving the efficiency of encryption schemes. We adapt the hybrid encryption approach to the setting of functional encryption via a technique for embedding a “hidden execution thread” in the decryption keys of the underlying scheme, which will only be activated within the proof of security of the resulting scheme. As an additional application of this technique, we show how to construct functional encryption schemes for arbitrary circuits starting from ones for shallow circuits (NC1 or even TC0).
1 Introduction

Traditional notions of public-key encryption provide all-or-nothing access to data: owners of the secret key can recover the entire message from a ciphertext, whereas those who do not know the secret key learn nothing at all. Functional encryption, a revolutionary notion originating from the work of Sahai and Waters [SW05] and studied in several follow-up works ([KSW08, LOS+10, BSW11, O’N10] and many others), is a modern type of encryption scheme where the owner of the (master) secret key can release function-specific secret keys $sk_f$, referred to as functional keys, which enable a user holding an encryption of a message $x$ to compute $f(x)$ but nothing else. Intuitively, in terms of indistinguishability-based security, encryptions of any two messages, $x_0$ and $x_1$, should be computationally indistinguishable given access to functional keys for any function $f$ such that $f(x_0) = f(x_1)$.

While initial constructions of functional encryption schemes [BF03, BCOP04, KSW08, LOS+10] were limited to restricted function classes such as point functions and inner products, recent developments have dramatically improved the state of the art. In particular, the works of Sahai and Seyalioglu [SS10] and Gorbunov, Vaikuntanathan and Wee [GVW12] showed that a scheme supporting a single functional key can be based on any semantically-secure encryption scheme. This result can be extended to the case where the number of functional keys is polynomial and known a-priori [GVW12]. Goldwasser, Kalai, Popa, Vaikuntanathan and Zeldovich [GKP+13] constructed a scheme with succinct ciphertexts based on a specific hardness assumption (Learning with Errors).

The first functional encryption scheme that supports a-priori unbounded number of functional keys was constructed by Garg, Gentry, Halevi, Raykova, Sahai and Waters [GGH+13], based on the existence of a general-purpose indistinguishability obfuscator (for which a heuristic construction is presented in the same paper). Garg et al. showed that given any such obfuscator, their functional encryption scheme is selectively secure. At a high level, selective security guarantees security only for messages that are fixed ahead of time (i.e., before the adversary even interacts with the system). Whereas security only for such messages may be justified in some cases, it is typically too restrictive for realistic applications. A more realistic notion is that of adaptive security (often called full security), which guarantees security even for messages that can be adaptively chosen at any point in time.

Historically, the first functional encryption schemes were only proven selectively secure [BB04, GPSW06, KSW08, GVW13, GKP+13]. The problem of constructing adaptively secure schemes seems significantly more challenging and only few approaches are known. A simple observation is that if a selectively-secure scheme’s message space is not too large, e.g., $\{0, 1\}^n$ for a relatively small $n$, then any adaptively-chosen message $x$ can be guessed ahead of time with probability $2^{-n}$. Starting with a sub-exponential hardness assumption, and taking the security parameter to be polynomial in $n$ allows us to argue that the selectively-secure scheme is in fact also adaptively secure. This observation is known as “complexity leveraging” and is clearly not satisfactory in general.

The powerful “dual system encryption” method, introduced by Waters [Wat09], has been used to construct adaptively-secure attribute-based encryption scheme (a restricted notion of functional encryption) for formulas, as well as an adaptively-secure functional encryption scheme for linear functions [LOS+10]. However, this method is a general outline, and each construction was so far required to tailor the solution based on its specialized assumption. In some cases, such as attribute-based encryption for circuits, it is still not known how to implement dual system encryption to achieve adaptive security (although Garg, Gentry, Halevi and Zhandry [GGHZ14a] show how to do this with custom-built methods and hardness assumptions).

Starting with [GGH+13], there has been significant effort in the research community to construct an adaptively-secure general-purpose functional encryption scheme with an unbounded number of functional keys. Boyle, Chung and Pass [BCP14] constructed an adaptively secure scheme, under the assumption that differing-input obfuscators exist (these are stronger primitives than the indistinguishability obfuscators used by [GGH+13]). Following their work, Waters [Wat14] and Garg, Gentry, Halevi and Zhandry [GGHZ14b] constructed specific adaptively-secure schemes assuming indistinguishability obfuscation and assuming non-standard assumptions on multilinear maps, respectively. Despite this significant progress, each of these constructions relies on somewhat tailored methods and techniques.
1.1 Our Results: From Selective to Adaptive Security

We show that any selectively secure functional encryption scheme implies an adaptively secure scheme, without relying on any additional assumptions. Our transformation applies equally to symmetric-key and public-key functional encryption, where the resulting adaptive scheme inherits its symmetric-key or public-key properties from the building block scheme. The following theorem summarizes our main contribution.

**Theorem 1** (informal). *Given any public-key (respectively symmetric-key) selectively-secure functional encryption scheme, there exists an adaptively secure public-key (respectively symmetric-key) functional encryption scheme supporting the class of bounded polynomial size circuits.*

[Vinod’s Note: Bounded poly size seems incorrect. We do arbitrary poly size.]

We require that the selective scheme supports a sufficiently rich function class. In particular it will be applied to functions with a-priori bounded polynomial size and a-priori bounded depth (essentially the depth of computing a weak pseudorandom function, which is logarithmic in the security parameter under mild assumptions). The resulting adaptively secure scheme, however, does not have a depth bound. Therefore a corollary of our construction is that selectively secure functional encryption with depth bound implies adaptively secure functional encryption without a depth bound.

Our transformation can be applied, for example, to the selectively-secure scheme of [GGH +13], or to the first simpler scheme in [Wat14], resulting in an adaptively secure functional encryption scheme based on indistinguishability obfuscation (and one-way functions). (Waters [Wat14] is able to construct an adaptively secure scheme under the same assumption, but using specific ad-hoc techniques and requiring significantly more complicated techniques.)

We view the significance of our result in a number of dimensions. First of all, it answers the basic call of cryptographic research to substantiate the existence of complicated primitives on that of simple primitives. We feel that this is of special interest in the case of adaptive security where it seemed that ad-hoc methods were required. Secondly, our construction, being of fairly low overhead, will allow to focus the attention of the research community in studying selectively secure functional encryption, rather than investing unwarranted efforts in the study of adaptive security. Lastly, we hope that our methods can be extended and employed towards those variants for which adaptive security is yet unattained generically, such as attribute based encryption for all polynomial size circuits.

1.2 Our Techniques

Our result is achieved by incorporating a number of techniques which will be explained in this section. In a nutshell, our main observation is that hybrid encryption (a.k.a key encapsulation) can be employed in the context of functional encryption, and has great potential in going from selective to adaptive security of encryption schemes. At a first glance, hybrid functional encryption should lead to a selective-to-adaptive transformation, given an additional weak component: A symmetric FE which is adaptively secure when only a single message query is allowed. We show that the latter can be constructed from any one-way function as a corollary of [GVW12, BS15]. However, the intuitive reasoning fails to translate into a proof of security. To resolve this issue, we use a technique we call The Trojan Method, which originates from De Caro et al.’s “trapdoor circuits” [CLJ +13] (similar ideas had been since used by Gentry et al. [GHRW14] and Brakerski and Segev [BS15]). We conclude this section with a short comparison of our technique with the aforementioned “dual system encryption” technique that had been used to achieve adaptively secure attribute based encryption.

**Hybrid Functional Encryption.** Hybrid encryption is a veteran technique in cryptography and has been used in a variety of settings. We show that in the context of functional encryption it is especially powerful.

The idea in hybrid encryption is to combine two encryption schemes: An “external” scheme (sometimes called KEM – Key Encapsulation Mechanism) and an “internal” scheme (sometimes called DEM – Data Encapsulation Mechanism). In order to encrypt a message in the hybrid scheme, a fresh key is generated for
the internal scheme, and is used to encrypt the message. Then the key itself is encrypted using the external scheme. The final hybrid ciphertext contains the two ciphertexts: $(\text{Enc}_{\text{ext}}(k), \text{Enc}_{\text{int},k}(m))$ (all external ciphertexts use the same key). To decrypt, one first decrypts the external ciphertext, retrieves $k$ and applies it to the internal ciphertext. Note that if, for example, the external scheme is public-key and the internal is symmetric key, then the resulting scheme will also be public key. Hybrid encryption is often used in cases where the external scheme is less efficient (e.g. in encrypting long messages) and thus there is an advantage in using it to encrypt only a short key, and encrypt the long message using the more efficient internal scheme. Lastly, note that the internal scheme only needs to be able to securely encrypt a single message.

The intuition as to why hybrid encryption may be good for achieving adaptive security is that the external scheme only encrypts keys for the internal scheme. Namely, it only encrypts messages from a predetermined and known distribution, so selective security should be enough for the external scheme. The hardness of adaptive security is “pushed” to the internal scheme, but there the task is easier since the internal scheme only needs to be able to encrypt a single message, and it can be private-key rather than public-key.

Let us see how to employ this idea in the case where both the internal and external schemes are FE schemes. To encrypt, we will generate a fresh master secret key for the internal scheme, and encrypt it under the external scheme. To generate a key for the function $f$, the idea is to generate a key for the function $G_f(\text{msk}_{\text{int}})$ which takes a master key for the internal scheme, and outputs a secret key for function $f$ under the internal scheme, using $\text{msk}_{\text{int}}$ (randomness is handled using a PRF). This will allow to decrypt in a two-step process as above. First apply the external secret-key for $G_f$ to the external ciphertext, this will give you an internal secret key for $f$, which is in turn applied to the internal ciphertext to produce $f(x)$.

For the external scheme, we will use a selectively secure FE scheme (for the sake of concreteness, let us say public-key FE). As explained above, selective security is sufficient here since all the messages encrypted using the external scheme can be generated ahead of time (i.e. they do not depend on the actual $x$’s that the user wishes to encrypt).

For the internal scheme, we require an FE scheme that is adaptively secure, but only supports the encryption of a single message. Fortunately, such a primitive can be derived from the works of [GVW12, BS15]. In [GVW12], the authors present an adaptively secure one-time bounded FE scheme. This scheme allows to only generate a key for one function, and to encrypt as many messages as the user wishes. This construction is based on the existence of semantically secure encryption, so the public-key version needs public-key encryption and the symmetric version needs symmetric encryption. While this primitive seems dual to what we need for our purposes, [BS15] shows how to transform private-key FE schemes into function private FE. In function-private FE, messages and functions enjoy the same level of privacy, in the sense that a user that produces $x_0, x_1, f_0, f_1$ such that $f_0(x_0) = f_1(x_1)$ cannot distinguish between $(\text{Enc}(x_0), \text{sk}_{f_0})$ and $(\text{Enc}(x_1), \text{sk}_{f_1})$. Therefore, after applying the [BS15] transformation, we can switch the roles of the functions and messages, and obtain a symmetric FE scheme which is adaptively secure for a single message and many functions. (We note that the symmetric version of the [GVW12] scheme can be shown to be function private even without the [BS15] transformation, however since this claim is not made explicitly in the paper we choose not to rely on it.)

Whereas intuitively this should solve the problem, it is not clear how to prove security of the new construction. In particular, we need to use the security of the internal scheme, while preserving the correct functionality of the function-ciphertext pair. This seems contradictory as for the former $\text{msk}_{\text{int}}$ must not be used in the reduction (otherwise the security of the internal scheme cannot be used), and for the latter $\text{msk}_{\text{int}}$ is needed, since otherwise $G_f$ will not be able to produce $\text{sk}_f$. To resolve this apparent contradiction, we use the Trojan method. [Vinod’s Note: This is a confusing paragraph. I am not sure I understand.]

**The Trojan Method.** The Trojan Method, which is a generalization of techniques used in [CIJ+13] and later in [GHRW14, BS15], is a way to embed a hidden functionality thread in an FE secret-key that can only be invoked by special ciphertexts generated using special (secret) back-door information. This thread remains completely unused in the normal operation of the scheme (and can be instantiated with meaningless functionality). In the proof, however, the secret thread will be activated by the challenge ciphertext in such a way that is indistinguishable to the user (= attacker). Namely, the user will not be able to tell that it is
executing the secret thread and not the main thread. This will be extremely beneficial to prove security. We wish to argue that in the view of the user, the execution of the main thread does not allow to distinguish between the encryption of two messages $x_0, x_1$. The problem is that for functionality purposes, the main thread has to know which input it is working on. This is where the hidden thread comes into the play. We will design the hidden thread so that in the eyes of the user, it is computationally indistinguishable from the main thread on the special messages $x_0, x_1$. However, in the hidden thread, the output can be computed in a way that does not distinguish between $x_0$ and $x_1$ (either by a statistical or a computational argument), which will allow us to conclude that encryptions of $x_0, x_1$ are indistinguishable.

In particular, this method will resolve the aforementioned conundrum in our proof outline above. In the proof, we will use the Trojan method to embed a hidden thread in which $\text{msk}_\text{int}$ is not used at all, but rather $G_f$ produces a precomputed internal $\text{sk}_f$. This will allow us to remove $\text{msk}_\text{int}$ from the challenge ciphertext and use the security properties of the internal scheme to argue that a internal encryption of $x_0, x_1$ are identical so long as $f(x_0) = f(x_1)$.

We note that an important special case of the above outline is when the trojan thread is a constant function. This had been the case in [CIJ+13, GHRW14], and this is the case in this work as well. However, we emphasize that our description here allows for greater generality since we allow the trojan thread to implement functionality that depends on the input $x$. We feel that this additional power may be useful for future applications.

Technically, the hidden thread is implemented using (standard) symmetric-key encryption, which in turn can be constructed starting with any one-way function. In the functional secret-key generation process for a function $f$, the secret-key generation process will produce a symmetric-key ciphertext $c$ (which can just be encryption of 0 or another fixed message, since it only needs to have meaningful content in the security proof). It will then consider the function $G_{f,c}$ that takes as input a pair $(x, s)$, and first checks whether it can decrypt $c$ using $s$ as a symmetric key. If it cannot, then it just runs $f$ on $x$ and returns the output. If $s$ actually decrypts $c$, we consider $f^* = \text{Dec}_s(c)$ (i.e. $c$ encrypts a description of a function), and the output is the execution of $f^*(x)$. The value $c$ is therefore used as a Trojan Horse: Its contents are hidden from the users of the scheme, however given a hidden command (in the form of the symmetric $s$) it can embed functionality that “takes over” the functional secret-key.

We note that in order to support the Trojan method, the decryption keys of our FE scheme need to perform symmetric decryption, branch operations, and execution of the function $f^*$. Thus we need to start with an FE scheme which allows for the generation of sufficiently expressive keys.

Our Trojan method can be seen as a weak form of function privacy in FE, but one that can be applied even in the context of public-key FE. In essence, we cannot hide the main thread of the evaluated function (this is unavoidable in public-key FE). However, we can hide the secret thread and thus allow the function to operate in a designated way for specially generated ciphertexts. (This interpretation is not valid for previous variants of this method such as “trapdoor circuits” [CIJ+13].)

[Vinod’s Note: I am 50-50 about keeping the shallow-to-deep versus removing. But I’d like to keep the dual system paragraph.]

For a fairly simple example on how the Trojan method is applied, see the outline for our shallow-to-deep transformation below.

**Example: Shallow-to-Deep FE.** We use the Trojan method to show that FE that only supports secret-keys for functions with shallow circuits (e.g. logarithmic depth) implies a scheme that works for circuits of arbitrary depth (although with a size bound). This construction is quite similar to [GHRW14, Appendix D].

The idea is fairly natural: Instead of producing a secret-key for the function, we produce a key for a shallow randomized encoding (RE) of the function (in particular, we use garbled circuits). A (computational) randomized encoding [IK00, AIK05] is a randomized process such that given $f, x$, $\text{RE}(f, x) = \text{RE}(f, x; \text{rand})$ is a randomized function that can be computed by a shallow circuit (assuming weak PRFs can be computed by shallow circuits). Furthermore, there is an efficient way to retrieve $f(x)$ from $\text{RE}(f, x)$, however nothing except $f(x)$ is revealed. The latter is guaranteed by the existence of a simulator such that $\text{Sim}(f(x)) \cong \text{RE}(f, x)$. We note that a similar approach had been attempted to leverage the supported depth of obfuscators
[GVW12, App14], however it appears to only work for strong notions of obfuscation such as virtual-black-box that are impossible to achieve in general.

Using the Trojan method, we show how this can be achieved for FE. As mentioned above, whenever the scheme asks for a secret-key for $f$, it will in fact receive a secret-key for a function $G$ that given $x$, outputs $G_{f,x}(x, k) = \text{RE}(f, x; \text{PRF}_{k}(t))$. To encrypt a value $x$, we choose $k$ at random and encrypt the pair $(x, k)$. The decryption process will include applying the functional secret key to obtain $\text{RE}(f, x)$ and computing $f(x)$ from that value. For security we have to show that if $x_0, x_1$ are such that $f(x_0) = f(x_1)$ then their encryptions are indistinguishable. This is where the Trojan comes into the play. Consider, for starters, the case where $x_0$ had been encrypted (with some PRF seed $k$). We will insert a Trojan thread into $G_{f,t}$, that, once activated, simply outputs $\text{RE}(f, x_0; \text{PRF}_{k^*}(t))$. Now, the hidden thread outputs exactly the same as the main thread, and therefore switching to the hidden thread will go unnoticed by the user (note that the hidden thread is only activated for the special encryption of $(x_0, k^*)$). Once the switch had been made, $x_0, k^*$ no longer need to appear in the ciphertext (since all the information about them had been embedded in the hidden thread). The hidden thread is indistinguishable from just outputting $\text{RE}(f, x_0; \text{rand})$, which is in turn indistinguishable from $\text{Sim}(f(x_0))$, which is in turn identical to $\text{Sim}(f(x_1))$. It follows that the user cannot distinguish between the case where $x_0$ had been encrypted and the case where $x_1$ had been encrypted.

**Relation to Dual-System Encryption.** Our approach takes some resemblance to the “Dual-System Encryption” method of Waters [Wat09] and followup works [LW10, LW12]. This method had been used to prove adaptive security for Identity Based Encryption and Attribute Based Encryption, based on the hardness of some problems on groups with bilinear-maps. In broad terms, in their proof the distribution of the ciphertext is changed into “semi-functional” mode in a way that is indiscernible by an observer. A semi-functional ciphertext is still decryptable by normal secret keys. Then, the secret-keys are modified into semi-functional form, which is useless in decrypting semi-functional ciphertexts. This is useful since in IBE and ABE, the challenge ciphertext is not supposed to be decryptable by those keys given to the adversary. Still, a host of algebraic techniques are used to justify the adversary’s inability to produce other semi-functional ciphertexts in addition to the challenge, which would foil the reduction.

Our proof technique also requires changing the distributions of the keys and challenge ciphertext. However, there are also major differences. Our modified ciphertext is not allowed to interact with properly generated secret keys, and therefore the distinction between “normal” and “semi-functional” does not fit here. Furthermore, in Identity Based and Attribute Based Encryption, the attacker in the security game is not allowed to receive keys that reveal any information on the message, which allows to generate semi-functional ciphertexts that do not contain any information, whereas in our case, there is a structured and well-defined output for any ciphertext and any key. This means that the information required for decryption (which can be a-priori unbounded) needs to be embedded in the keys. Lastly, our proof is completely generic and does not rely on the algebraic structure of the underlying hardness assumption as in previous implementations of this method.

2 Preliminaries

In this section we present the notation and basic definitions that are used in this work. For a distribution $X$ we denote by $x \leftarrow X$ the process of sampling a value $x$ from the distribution $X$. Similarly, for a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value $x$ from the uniform distribution over $\mathcal{X}$. For a randomized function $f$ and an input $x \in \mathcal{X}$, we denote by $y \leftarrow f(x)$ the process of sampling a value $y$ from the distribution $f(x)$. A function $\negl : \mathbb{N} \rightarrow \mathbb{R}$ is **negligible** if for any polynomial $p(\lambda)$ it holds that $\negl(\lambda) < 1/p(\lambda)$ for all sufficiently large $\lambda \in \mathbb{N}$.

2.1 Pseudorandom Functions and Symmetric Encryption

**Pseudorandom functions.** We rely on the following standard notion of a pseudorandom function family [GGM86], asking that a pseudorandom function be computationally indistinguishable from a truly random
function via oracle access.

**Definition 1.** A family $F = \{ \text{PRF}_K : \{0,1\}^n \rightarrow \{0,1\}^m : K \in K \}$ of efficiently-computable functions is pseudorandom if for every PPT adversary $A$ there exists a negligible function $\text{negl}(\cdot)$ such that

$$\left| \Pr_{K \leftarrow \mathcal{K}} [A^{\text{PRF}_K} (1^\lambda) = 1] - \Pr_{R \leftarrow \mathcal{U}} [A^R (1^\lambda) = 1] \right| \leq \text{negl}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$, where $\mathcal{U}$ is the set of all functions from $\{0,1\}^n$ to $\{0,1\}^m$.

We say that a pseudorandom function family $F$ is implementable in NC$^2$ if every function in $F$ can be implemented by a circuit of depth $c \cdot \log(n)$, for some constant $c$. We also consider the notion of a weak pseudorandom function family, asking that the above definition holds for adversaries that may access the functions on random inputs (that is, the oracles PRF$_K(\cdot)$ and $R(\cdot)$ take no input, and on each query they sample a uniform input $r$ and output PRF$_K(r)$ and $R(r)$, respectively).

**Symmetric encryption with pseudorandom ciphertexts.** A symmetric encryption scheme consists of a tuple of PPT algorithms $(\text{Sym.Setup}, \text{Sym.Enc}, \text{Sym.Dec})$. The algorithm $\text{Sym.Setup}$ takes as input a security parameter $\lambda$ in unary and outputs a key $K_{\lambda}$. The encryption algorithm $\text{Sym.Enc}$ takes as input a symmetric key $K_{\lambda}$ and a message $m$ and outputs a ciphertext $CT$. The decryption algorithm $\text{Sym.Dec}$ takes as input a symmetric key $K_{\lambda}$ and a ciphertext $CT$ and outputs the message $m$.

In this work, we require a symmetric encryption scheme II where the ciphertexts produced by $\text{Sym.Enc}$ are pseudorandom strings. Let $\text{OEnc}_{\lambda}(\cdot)$ denote the (randomized) oracle that takes as input a message $m$, chooses a random string $r$ and outputs $\text{Sym.Enc}(\lambda, m; r)$. Let $R_{\ell(\lambda)}(\cdot)$ denote the (randomized) oracle that takes as input a message $m$ and outputs a uniformly random string of length $\ell(\lambda)$ where $\ell(\lambda)$ is the length of the ciphertexts. More formally, we require that for every PPT adversary $A$ the following advantage is negligible in $\lambda$:

$$\text{Adv}_{\Pi,A}^{\text{symPR}}(\lambda) = \left| \Pr[A^{\text{OEnc}_{\text{symEnc}}(\cdot)}(1^\lambda) = 1] - \Pr[A^{R_{\ell(\lambda)}}(1^\lambda) = 1] \right|$$

where the probability is taken over the choice of $\text{Sym.K} \leftarrow \text{Sym.Setup}(1^\lambda)$, and over the internal randomness of $A$, $\text{OEnc}$ and $R_{\ell(\lambda)}$.

We note that such a symmetric encryption scheme with pseudorandom ciphertexts can be constructed from one-way functions, e.g. using weak pseudorandom functions by defining $\text{Sym.Enc}(\lambda, m; r) = (r, \text{PRF}_K(r) \oplus m)$ (see [Gol04] for more details).

### 2.2 Public-Key Functional Encryption

A public-key functional encryption (FE) scheme $\Pi_{\text{Pub}}$ over a message space $M = \{M_\lambda\}_{\lambda \in \mathbb{N}}$ and a function space $F = \{F_\lambda\}_{\lambda \in \mathbb{N}}$ is a tuple $(\text{Pub.Setup}, \text{Pub.KeyGen}, \text{Pub.Enc}, \text{Pub.Dec})$ of PPT algorithms with the following properties:

- **Pub.Setup($1^\lambda$):** The setup algorithm takes as input the unary representation of the security parameter, and outputs a public key MPK and a secret key MSK.

- **Pub.KeyGen(MSK, $f$):** The key-generation algorithm takes as input a secret key MSK and a function $f \in F_\lambda$, and outputs a functional key $sk_f$.

- **Pub.Enc(MPK, $m$):** The encryption algorithm takes as input a public key MPK and a message $m \in M_\lambda$, and outputs a ciphertext $CT$.

- **Pub.Dec($sk_f$, $CT$):** The decryption algorithm takes as input a functional key $sk_f$ and a ciphertext $CT$, and outputs $m \in M_\lambda \cup \{\bot\}$. 

We say that such a scheme is defined for a complexity class \( C \) if it supports all the functions that can be implemented in \( C \). In terms of correctness, we require that there exists a negligible function \( \text{negl}(\cdot) \) such that for all sufficiently large \( \lambda \in \mathbb{N} \), for every message \( m \in M_\lambda \), and for every function \( f \in F_\lambda \) it holds that \( \Pr[\text{Pub.Dec(Pub.KeyGen(MSK, f), Pub.Enc(MPK, m))} = f(m)] \geq 1 - \text{negl}(\lambda) \), where \((\text{MPK, MSK}) \leftarrow \text{Pub.Setup}(1^\lambda)\), and the probability is taken over the random choices of all algorithms.

We consider the standard selective and adaptive indistinguishability-based notions for functional encryption (see, for example, [BSW11, O’N10]). Intuitively, these notions ask that encryptions of any two messages, \( m_0 \) and \( m_1 \), should be computationally indistinguishable given access to functional keys for any function \( f \) such that \( f(m_0) = f(m_1) \). In the case of selective security, adversaries are required to specify the two messages in advance (i.e., before interacting with the system). In the case of adaptive security, adversaries are allowed to specify the two messages even after obtaining the public key and functional keys.

**Definition 2** (Selective security). A public-key functional encryption scheme \( \Pi = (\text{Sel.Setup, Sel.KeyGen, Sel.Enc, Sel.Dec}) \) over a function space \( F = \{F_\lambda\}_{\lambda \in \mathbb{N}} \) and a message space \( M = \{M_\lambda\}_{\lambda \in \mathbb{N}} \) is selectively secure if for any PPT adversary \( A \) there exists a negligible function \( \text{negl}(\cdot) \) such that

\[
\text{Adv}_{\Pi, A}^{\text{Sel}}(\lambda) = \left| \Pr[\text{Expt}_{\Pi, A}^{\text{Sel}}(\lambda, 0) = 1] - \Pr[\text{Expt}_{\Pi, A}^{\text{Sel}}(\lambda, 1) = 1] \right| \leq \text{negl}(\lambda)
\]

for all sufficiently large \( \lambda \in \mathbb{N} \), where for each \( b \in \{0, 1\} \) and \( \lambda \in \mathbb{N} \) the experiment \( \text{Expt}_{\Pi, A}^{\text{Sel}}(\lambda, b) \), modeled as a game between the adversary \( A \) and a challenger, is defined as follows:

1. **Setup phase**: The challenger samples \((\text{Sel.MPK, Sel.MSK}) \leftarrow \text{Sel.Setup}(1^\lambda)\).
2. **Challenge phase**: On input \( 1^\lambda \) the adversary submits \((m_0, m_1)\), and the challenger replies with \((\text{Sel.MPK and CT}) \leftarrow \text{Sel.Enc(Sel.MPK, m_k)}\).
3. **Query phase**: The adversary adaptively queries the challenger with any function \( f \in F_\lambda \) such that \( f(m_0) = f(m_1) \). For each such query, the challenger replies with \( \text{Sel.sk}_f \leftarrow \text{Sel.KeyGen(Sel.MSK, f)} \).
4. **Output phase**: The adversary outputs a bit \( b' \) which is defined as the output of the experiment.

**Definition 3** (Adaptive security). A public-key functional encryption scheme \( \Pi = (\text{Ad.Setup, Ad.KeyGen, Ad.Enc, Ad.Dec}) \) over a function space \( F = \{F_\lambda\}_{\lambda \in \mathbb{N}} \) and a message space \( M = \{M_\lambda\}_{\lambda \in \mathbb{N}} \) is adaptively secure if for any PPT adversary \( A \) there exists a negligible function \( \text{negl}(\cdot) \) such that

\[
\text{Adv}_{\Pi, A}^{\text{Ad}}(\lambda) = \left| \Pr[\text{Expt}_{\Pi, A}^{\text{Ad}}(\lambda, 0) = 1] - \Pr[\text{Expt}_{\Pi, A}^{\text{Ad}}(\lambda, 1) = 1] \right| \leq \text{negl}(\lambda)
\]

for all sufficiently large \( \lambda \in \mathbb{N} \), where for each \( b \in \{0, 1\} \) and \( \lambda \in \mathbb{N} \) the experiment \( \text{Expt}_{\Pi, A}^{\text{Ad}}(1^\lambda, b) \), modeled as a game between the adversary \( A \) and a challenger, is defined as follows:

1. **Setup phase**: The challenger samples \((\text{Ad.MPK, Ad.MSK}) \leftarrow \text{Ad.Setup}(1^\lambda)\), and sends \( \text{Ad.MPK} \) to the adversary.
2. **Query phase I**: The adversary adaptively queries the challenger with any function \( f \in F_\lambda \). For each such query, the challenger replies with \( \text{Sel.sk}_f \leftarrow \text{Ad.KeyGen(Ad.MSK, f)} \).
3. **Challenge Phase**: The adversary submits \((m_0, m_1)\) such that \( f(m_0) = f(m_1) \) for all function queries \( f \) made so far, and the challenger replies with \( \text{CT} \leftarrow \text{Ad.Enc(Ad.MSK, m_k)} \).
4. **Query phase II**: The adversary adaptively queries the challenger with any function \( f \in F_\lambda \) such that \( f(m_0) = f(m_1) \). For each such query, the challenger replies with \( \text{Sel.sk}_f \leftarrow \text{Ad.KeyGen(Ad.MSK, f)} \).
5. **Output phase**: The adversary outputs a bit \( b' \) which is defined as the output of the experiment.

\(^1\)Our notions of security consider a single challenge, and in the public-key setting these are known to be equivalent to their multi-challenge variants via a standard hybrid argument.
3 Our Transformation in the Public-Key Setting

In this section we present our transformation from selective security to adaptive security for public-key functional encryption schemes. In addition to any selectively-secure public-key functional encryption scheme (see Definition 2), our transformation requires a private-key functional encryption scheme that is adaptively-secure for a single message query and many function queries. Based on [GVW12, BS15], such a scheme can be based on any one-way function².

More specifically, we rely on the following building blocks (all of which are implied by any selectively-secure public-key functional encryption scheme):


2. An adaptively-secure single-ciphertext private-key functional encryption scheme³ $\text{OneCT} = (\text{OneCT.Setup}, \text{OneCT.KeyGen}, \text{OneCT.Enc}, \text{OneCT.Dec})$.


4. A pseudorandom function family $\mathcal{F}$ with a key space $\mathcal{K}$.

Our adaptively-secure scheme $\text{Ad} = (\text{Ad.Setup}, \text{Ad.KeyGen}, \text{Ad.Enc}, \text{Ad.Dec})$ is defined as follows.

- **The setup algorithm:** On input $1^\lambda$ the algorithm $\text{Ad.Setup}$ samples $(\text{Sel.MPK}, \text{Sel.MSK}) \leftarrow \text{Sel.Setup}(1^\lambda)$, and outputs $\text{Ad.MPK} = \text{Sel.MPK}$ and $\text{Ad.MSK} = \text{Sel.MSK}$.

- **The key-generation algorithm:** On input the secret key $\text{Ad.MSK} = \text{Sel.MSK}$ and a function $f$, the algorithm $\text{Ad.KeyGen}$ first samples $C_E \leftarrow \{0,1\}^{E(\lambda)}$ and $\tau \leftarrow \{0,1\}^{\tau(\lambda)}$ uniformly and independently. Then, it computes and outputs $A_d.sk_f = \text{Sel.sk}_G \leftarrow \text{Sel.KeyGen}(\text{Sel.MSK}, G_{f,C_E,\tau})$, where the function $G_{f,C_E,\tau}$ is defined in Figure 1.

$G_{f,C_E,\tau}(\text{OneCT.SK}, K, \text{Sym.K}, \beta)$:

1. If $\beta = 1$ output $\text{OneCT.sk}_f \leftarrow \text{Sym.Dec}(\text{Sym.K}, C_E)$.
2. Otherwise, output $\text{OneCT.sk}_f \leftarrow \text{OneCT.KeyGen}(\text{OneCT.SK}, f; \text{PRF}(\tau))$.

**Figure 1** The function $G_{f,C_E,\tau}$.

- **The encryption algorithm:** On input the public key $\text{Ad.MPK} = \text{Sel.MPK}$ and a message $m$, the algorithm $\text{Ad.Enc}$ first samples $K \leftarrow K_\lambda$ and $\text{OneCT.SK} \leftarrow \text{OneCT.Setup}(1^\lambda)$. Then, it outputs $\text{CT} = (CT_0, CT_1)$, where

  $CT_0 \leftarrow \text{OneCT.Enc}(\text{OneCT.SK}, m)$

  $CT_1 \leftarrow \text{Sel.Enc}(\text{Sel.MPK}, (\text{OneCT.SK}, K, 0^λ, 0))$.

- **The decryption algorithm:** On input a functional key $A_d.sk_f = \text{Sel.sk}_G$ and a ciphertext $\text{CT} = (CT_0, CT_1)$, the decryption algorithm $\text{Ad.Dec}$ first computes $\text{OneCT.sk}_f \leftarrow \text{Sel.Dec}(\text{Sel.sk}_G, CT_1)$. Then, it computes $m \leftarrow \text{OneCT.Dec}(\text{OneCT.sk}_f, CT_0)$ and outputs $m$.

²Gorbunov et al. [GVW12] constructed a private-key functional encryption scheme that is adaptively secure for a single function query and many message queries based on any private-key encryption scheme (and thus based on any one-way function). Any such scheme can be turned into a function private one using the generic transformation of Brakerski and Segev [BS15], and then one can simply switch the roles of functions and messages [AAB13, BS15]. This results in a private-key scheme that is adaptively secure for a single message query and many function queries.

³That is, a private-key functional encryption scheme that is adaptively-secure for a single message query and many function queries (as discussed above).
The correctness of the above scheme easily follows from that of its underlying building blocks, and in the remainder of this section we prove the following theorem:

**Theorem 2.** Assuming that: (1) Sel is a selectively-secure public-key functional encryption scheme, (2) OneCT is an adaptively-secure single-ciphertext private-key functional encryption scheme, (3) SYM is a symmetric encryption scheme with pseudorandom ciphertexts, and (4) $F$ is a pseudorandom function family, then Ad is an adaptively-secure public-key functional encryption scheme.

**Proof.** We show that any PPT adversary $A$ succeeds in the adaptive security game (see Definition 3) with only negligible probability. We will show this in a sequence of hybrids. We denote the advantage of the adversary in Hybrid$_{1,b}$ to be the probability that the adversary outputs 1 in this hybrid and this quantity is denoted by $Adv^A_{1,b}$. For $b \in \{0,1\}$, we define the following hybrids.

**Hybrid$_{1,b}$:** This corresponds to the real experiment when the challenger encrypts the message $m_b$. More precisely, the challenger produces an encryption $CT = (CT_0, CT_1)$ where

$$CT_0 \leftarrow \text{OneCT.Enc}(\text{OneCT.SK}, m) \quad \text{and} \quad CT_1 \leftarrow \text{Sel.Enc}(\text{Sel.MPK}, (\text{OneCT.SK}, K, 0^\lambda, 0)).$$

**Hybrid$_{2,b}$:** The challenger replaces the hard-coded ciphertext $C_E$ in every functional key corresponding to a query $f$ made by the adversary, with a symmetric key encryption of OneCT.sk$_f$ (note that each key has its own different $C_E$). Here, OneCT.sk$_f$ is the output of OneCT.KeyGen(OneCT.SK*, $f$; PRF$_K^*(\tau)$) and $K^*$ is a PRF key drawn from the key space $K$. Further, the symmetric encryption is computed with respect to Sym.$K^*$, where Sym.$K^*$ is the output of Sym.Setup(1$^\lambda$) and $\tau$ is the tag associated to the functional key of $f$. The same Sym.$K^*$ and $K^*$ are used while generating all the functional keys, and $K^*$ is used for generating the challenge ciphertext $CT^* = (CT_0^*, CT_1^*)$ (that is, $CT_0^* \leftarrow \text{OneCT.Enc}(\text{OneCT.SK}^*, m_b)$ and $CT_1^* \leftarrow \text{Sel.Enc}(\text{Sel.MSK}, (\text{OneCT.SK}^*, K^*, 0^\lambda, 0)))$. The rest of the hybrid is the same as the previous hybrid, Hybrid$_{1,b}$.

Note that the symmetric key Sym.$K^*$ is not used for any purpose other than generating the values $C_E$. Therefore, the pseudorandom ciphertexts property of the symmetric scheme implies that Hybrid$_{2,b}$ and Hybrid$_{1,b}$ are indistinguishable.

**Claim 1.** Assuming the pseudorandom ciphertexts property of SYM, for each $b \in \{0,1\}$ we have $|Adv^A_{1,b} - Adv^A_{2,b}| \leq \text{negl}(\lambda)$.

**Proof.** Suppose there exists an adversary such that the difference in the advantages is non-negligible, then we construct a reduction that can break the security of SYM. The reduction internally executes the adversary by simulating the role of the challenger in the adaptive public-key FE game. It answers both the message and the functional queries made by the adversary as follows. The reduction first executes OneCT.Setup(1$^\lambda$) to obtain OneCT.SK*. It then samples $K^*$ from $K$. Further, the reduction generates Sel.MSK, which is the output of Sel.Setup(1$^\lambda$) and Sym.$K^*$, which is the output of Sym.Setup(1$^\lambda$). When the adversary submits a functional query $f$, the reduction first picks $\tau$ at random. The reduction executes OneCT.KeyGen(OneCT.SK*, $f$; PRF($K^*(\tau)$)) to obtain OneCT.sk$_f$. It then sends OneCT.sk$_f$ to the challenger of the symmetric encryption scheme. The challenger returns back with $C_E$, where $C_E$ is either a uniformly random string or it is an encryption of OneCT.sk$_f$. The reduction then generates a selectively-secure FE functional key of $G_{f,C_E,\tau}$ and denote the result by Sel.sk$_G$ which is sent to the adversary. The message queries made by the adversary are handled as in Hybrid$_1$. That is, the adversary submits the message-pair query $(m_0, m_1)$ and the reduction sends $CT^* = (CT_0^*, CT_1^*)$ back to the adversary, where $CT_0^* = \text{OneCT.Enc}(\text{OneCT.SK}^*, m_b)$ and $CT_1^* = \text{Sel.Enc}(\text{Sel.MSK}, (0^\lambda, 0^\lambda, \text{Sym.$K^*$}, 1))$.

If the challenger of the symmetric key encryption scheme sends a uniformly random string back to the reduction every time the reduction makes a query to the challenger then we are in Hybrid$_{1,b}$, otherwise we are in Hybrid$_{2,b}$. Since the adversary can distinguish both the hybrids with non-negligible probability, we have that the reduction breaks the security of the symmetric key encryption scheme with non-negligible probability. From our hypothesis, we have that the reduction breaks the security of the symmetric key encryption scheme with non-negligible probability. This proves the claim. \qed
Hybrid\textsubscript{3,b}: The challenger modifies the challenge ciphertext $CT^* = (CT_0^*, CT_1^*)$ so that $CT_1^*$ is an encryption of $(0^λ, 0^λ, Sym.K^*, 1)$. The ciphertext component $CT_0^*$ is not modified (i.e., $CT_0^* = \mathrm{OneCT.Enc}(OneCT.SK^*, m_0)$). The rest of the hybrid is the same as the previous hybrid, Hybrid\textsubscript{2,b}.

Note that the functionality of the functional keys generated using the underlying selectively-secure scheme is unchanged with the modified $CT_1^*$. Therefore, its selective security implies that Hybrid\textsubscript{3,b} and Hybrid\textsubscript{2,b} are indistinguishable.

Claim 2. Assuming the selective security of Sel, for each $b \in \{0, 1\}$ we have $|\mathrm{Adv}\textsubscript{3,b}^A - \mathrm{Adv}\textsubscript{2,b}^A| \leq \text{negl}(\lambda)$.

Proof. Suppose the claim is not true for some adversary $A$, we construct a reduction that breaks the security of Sel. Our reduction will internally execute $A$ by simulating the role of the challenger of the adaptive FE game.

Our reduction first executes $\text{OneCT.Setup}(1^λ)$ to obtain $\text{OneCT.SK}^*$. It then samples $K^*$ from $K$. It also executes $\text{Sym.Setup}(1^λ)$ to obtain $\text{Sym.K}^*$. The reduction then sends the message pair $((\text{OneCT.SK}^*, K^*, 0^λ, 0)$, $(0^λ, 0^λ, \text{Sym.K}^*, 1))$ to the challenger of the selective game. The challenger replies back with the public key $\text{Sel.MPK}$ and the challenge ciphertext $CT_1^*$. The reduction is now ready to interact with the adversary $A$. If $A$ makes a functional query $f$, then the reduction constructs the circuit $G_{f,C_E,τ}$ as in Hybrid\textsubscript{2,b}. It then queries the challenger of the selective game with the function $G$ and in return it gets the key $\text{Sel.sk}_G$. The reduction then sets $\text{Ad.sk}_f$ to be $\text{Sel.sk}_G$ which it then sends back to $A$. If $A$ submits a message pair $(m_0, m_1)$, the reduction executes $\text{OneCT.Enc}(\text{OneCT.SK}^*, m_0)$ to obtain $CT_0^*$. It then sends the ciphertext $CT^* = (CT_0^*, CT_1^*)$ to the adversary. The output of the reduction is the output of $A$.

We claim that the reduction is a legal adversary in the selective security game of Sel, i.e., for challenge message query $M_0 = (\text{OneCT.SK}^*, K^*, 0^λ, 0)$, $M_1 = (0^λ, 0^λ, \text{Sym.K}^*, 1)$ and every functional query of the form $G_{f,C_E,τ}$ made by the reduction, we have that $G_{f,C_E,τ}(M_0) = G_{f,C_E,τ}(M_1)$: By definition, $G_{f,C_E,τ}(M_0)$ is the functional key of $f$, with respect to key $\text{OneCT.SK}^*$ and randomness $\text{PRF}_{K^*}(τ)$. Further, $G_{f,C_E,τ}(M_1)$ is the decryption of $C_E$ which is nothing but the functional key of $f$, with respect to key $\text{OneCT.SK}^*$ and randomness $\text{PRF}_{K^*}(τ)$. This proves that the reduction is a legal adversary in the selective security game.

If the challenger of the selective game sends back an encryption of $(\text{OneCT.SK}^*, K^*, 0^λ, 0)$ then we are in Hybrid\textsubscript{2,b} else if the challenger encrypts $(0^λ, 0^λ, \text{Sym.K}^*, 1)$ then we are in Hybrid\textsubscript{3,b}. By our hypothesis, this means the reduction breaks the security of the selective game with non-negligible probability that contradicts the security of Sel. This completes the proof of the claim.

Hybrid\textsubscript{4,b}: For every function query $f$ made by the adversary, the challenger generates $C_E$ by executing $\text{Sym.Enc}(\text{Sym.K}^*, \text{OneCT.sk}_f)$, with $\text{OneCT.sk}_f$ being the output of $\text{OneCT.KeyGen}(\text{OneCT.SK}^*, f; \text{R})$, where $R$ is picked at random. The rest of the hybrid is the same as the previous hybrid.

Note that the PRF key $K^*$ is not explicitly needed in the previous hybrid, and therefore the pseudorandomness of $F$ implies that Hybrid\textsubscript{4,b} and Hybrid\textsubscript{3,b} are indistinguishable.

Claim 3. Assuming that $F$ is a pseudorandom function family, for each $b \in \{0, 1\}$ we have $|\mathrm{Adv}\textsubscript{3,b}^A - \mathrm{Adv}\textsubscript{4,b}^A| \leq \text{negl}(λ)$.

Proof. Suppose the claim is false for some PPT adversary $A$, we construct a reduction that internally executes $A$ and breaks the security of the pseudorandom function family $F$. The reduction simulates the role of the challenger of the adaptive game when interacting with $A$. The reduction answers the functional queries, made by the adversary as follows; the message queries are answered as in Hybrid\textsubscript{3,b} (or Hybrid\textsubscript{4,b}). For every functional query $f$ made by the adversary, the reduction picks $τ$ at random which is then forwarded to the challenger of the PRF security game. In response it receives $R^*$. The reduction then computes $C_E$ to be $\text{Sym.Enc}(\text{Sym.K}^*, \text{OneCT.sk}_f)$, where $\text{OneCT.sk}_f = \text{OneCT.KeyGen}(\text{OneCT.SK}^*, f; \text{R})$. The reduction then proceeds as in the previous hybrids to compute the functional key $\text{Ad.sk}_f$ which it then sends to $A$.

If the challenger of the PRF game sent $R^* = \text{PRF}_{K^*}(τ)$ back to the reduction then we are in Hybrid\textsubscript{1,b} else if $R^*$ is generated at random by the challenger then we are in Hybrid\textsubscript{4,b}. From our hypothesis this means that the probability that the reduction distinguishes the pseudorandom value from random (at the point $τ$) is non-negligible, contradicting the security of the pseudorandom function family $F$. 

\hfill \Box
We now conclude the proof of the theorem by showing that $\text{Hybrid}_{4,0}$ is computationally indistinguishable from $\text{Hybrid}_{4,1}$ based on the adaptive security of the underlying single-ciphertext scheme.

**Claim 4.** Assuming the adaptive security of the scheme OneCT, we have $|\text{Adv}_{4,0}^A - \text{Adv}_{4,1}^A| \leq \text{neg}(\lambda)$.

**Proof.** Suppose there exists a PPT adversary $A$, such that the claim is false. We design a reduction $B$ that internally executes $A$ to break the adaptive security of OneCT.

The reduction simulates the role of the challenger of the adaptive public-key FE game. It answers both the functional as well as message queries made by the adversary as follows. If $A$ makes a functional query $f$ then it forwards it to the challenger of the adaptively-secure single-ciphertext FE scheme. In return it receives $\text{OneCT}.sk_f$. It then encrypts it using the symmetric encryption scheme, where the symmetric key is picked by the reduction itself, and denote the resulting ciphertext to be $C_E$. The reduction then constructs the circuit $G_{f,C_E,\tau}$, with $\tau$ being picked at random, as in the previous hybrids. Finally, the reduction computes the selective public-key functional key of $G_{f,C_E,\tau}$, where the reduction itself picks the master secret key of selective public-key FE scheme. The resulting functional key is then sent to $A$. If $A$ makes a message-pair query $(m_0, m_1)$, the reduction forwards this message pair to the challenger of the adaptive game. In response it receives $\text{CT}_0^\tau$. The reduction then generates $\text{CT}_1^\tau$ on its own where $\text{CT}_1^\tau$ is the selective FE encryption of $(0^\lambda, 0^\lambda, \text{Sym}.K^*, 1)$. The reduction then sends $\text{CT}^\tau = (\text{CT}_0^\tau, \text{CT}_1^\tau)$ to $A$. The output of the reduction is the output of $A$.

We note that the reduction is a legal adversary in the adaptive game of OneCT, i.e., for every challenge message query $(m_0, m_1)$, functional query $f$, we have that $f(m_0) = f(m_1)$: this follows from the fact that (i) the functional queries (resp., challenge message query) made by the adversary (of Ad) is the same as the same the functional queries (resp., challenge message query) made by the reduction, and (ii) the adversary (of Ad) is a legal adversary. This proves that the reduction is a legal adversary in the adaptive game.

If the challenger sends an encryption of $m_0$ then we are in $\text{Hybrid}_{4,0}$ and if the challenger sends an encryption of $m_1$ then we are in $\text{Hybrid}_{4,1}$. From our hypothesis, this means that the reduction breaks the security of OneCT. This proves the claim.

\[\Box\]

\[\Box\]

## 4 From Shallow Circuits to All Circuits

In this section we show that a functional encryption scheme that supports functions computable by shallow circuits can be transformed into one that supports functions computable by arbitrarily deep circuits. In particular, the shallow class can be any class in which weak pseudorandom functions can be computed and has some composition properties.\(^4\) For concreteness we consider here the class $\text{NC}^1$, which can compute weak pseudorandom functions under standard cryptographic assumptions such as DDH or LWE (a lower complexity class such as $\text{TC}^0$ is also sufficient under standard assumptions). We focus here on private-key functional encryption schemes, and note that an essentially identical transformation applies for public-key scheme.

While we present a direct reduction below, we notice that this property can be derived from the transformation in Section 3, by recalling some properties of Gurbunov et al.’s [GVW12] single-key functional encryption scheme. One can verify that their setup algorithm can be implemented in $\text{NC}^1$ (under the assumption that it can evaluate weak pseudorandom functions), regardless of the depth of the function being implemented. This property carries through even after applying the function privacy transformation of Brakerski and Segev [BS15]. Lastly, to implement our approach we need a symmetric encryption scheme with decryption in $\text{NC}^1$, which again translates to the evaluation of a weak pseudorandom function [NR04, BPR12].

\(^4\)Similarly to the class WEAK defined in [App14].
Randomized encodings [IK00]. A randomized encoding scheme for a function class $F$ consists of two PPT algorithms $(\text{RE.Encode, RE.Decode})$. The PPT algorithm RE.Encode takes as input $(1^\lambda, F, x, r)$, where $\lambda$ is the security parameter, $F : \{0, 1\}^\lambda \rightarrow \{0, 1\}$ is a function in $F$, instance $x \in \{0, 1\}^\lambda$ and randomness $r$. The output is denoted by $\hat{F}(x;r)$. The PPT algorithm RE.Decode takes as input $F(x;r)$ and outputs $y = F(x)$.

The security property states that there exists a PPT algorithm Sim that takes as input $(1^\lambda, F(x))$ and outputs $\text{SimOut}_F(x)$ such that any PPT adversary cannot distinguish the distribution $\{\text{SimOut}_F(x)\}$ from the distribution $\{\text{SimOut}_F(x)\}$. The following corollary is derived from applying Yao’s garbled circuit technique using a weak PRF based encryption algorithm.

Corollary 1. Assuming a family of weak pseudorandom functions that can be evaluated in NC$^1$, there exists a randomized encoding scheme $(\text{RE.Encode, RE.Decode})$ for the class of polynomial size circuits, such that RE.Encode is computable in NC$^1$.

Our transformation. Let $\mathcal{NC}\mathcal{FE} = (\text{NCFE.Setup, NCFE.KeyGen, NCFE.Enc, NCFE.Dec})$ be a private-key functional encryption scheme for the class NC$^1$. We assume that $\mathcal{NC}\mathcal{FE}$ supports functions with multi-bit outputs, otherwise it is always possible to produce a functional key for each output bit separately. We also use a pseudorandom function family denoted by $\mathcal{F} = \{\text{PRF}_K()\}_{K \in \mathbb{K}}$ and a symmetric encryption scheme $\text{SYM} = (\text{Sym.Setup, Sym.Enc, Sym.Dec})$. We construct a private-key functional encryption scheme $\mathcal{PF}\mathcal{E} = (\text{PFE.Setup, PFE.KeyGen, PFE.Enc, PFE.Dec})$ as follows.

- **The setup algorithm:** On input $1^\lambda$ the algorithm PFE.Setup samples and outputs $MSK \leftarrow \text{NCFE.Setup}(1^\lambda)$.

- **The key-generation algorithm:** On input the secret key $MSK$ and a circuit $F$, the algorithm PFE.KeyGen first samples $C_E \leftarrow \{0, 1\}^{\tau_F(\lambda)}$ and $\tau \leftarrow \{0, 1\}^\lambda$ uniformly and independently. Then, it computes a functional key $SK_G \leftarrow \text{NCFE.KeyGen}(MSK,G_{F,C_E,\tau})$, where the function $G_{F,C_E,\tau}$ is defined in figure 2, and outputs $(SK_G,F,C_E,\tau)$.

\[G_{f,C_E,\tau}(x, K_P, K_E, \beta):\]

1. If $\beta = 1$ output $\text{Sym.Dec}_{K_E}(C_E)$.
2. Otherwise, output $\hat{F}(x; \text{PRF}_{K_P}(\tau)) = \text{RE.Encode}(F, x; \text{PRF}_{K_P}(\tau))$.

$\text{Figure 2}$ The function $G_{f,C_E,\tau}$.

- **The encryption algorithm:** On input the secret key $MSK$ and a message $x$, the algorithm PFE.Enc first samples $K_P \leftarrow \{0, 1\}^\lambda$, and then computes and outputs $C \leftarrow \text{NCFE.Enc}(MSK, (x, K_P, 0^\lambda, 0))$.

- **The decryption algorithm:** On input a functional key $SK_F = (SK_G,F,C_E,\tau)$ and a ciphertext $C$, the decryption algorithm PFE.Dec computes $\hat{F}(x) \leftarrow \text{NCFE.Dec}(SK_G,F,C_E,\tau, C)$ and then outputs $\text{RE.Decode}(\hat{F}(x))$.

The correctness of the above scheme easily follows from that of its underlying building blocks, and in the remainder of this section we provide a sketch for proving the following theorem:

Theorem 3. Assuming that: (1) $\mathcal{NC}\mathcal{FE}$ is a selectively-secure private-key functional encryption scheme for NC$^1$, (2) $\text{SYM}$ is a symmetric encryption scheme with pseudorandom ciphertexts whose decryption circuit is in NC$^1$, (3) $\text{PRF}$ is a weak pseudorandom function family which can be evaluated in NC$^1$, and (4) $(\text{RE.Encode, RE.Decode})$ is a randomized encoding scheme with encoding in NC$^1$, then $\mathcal{PF}\mathcal{E}$ is a selectively-secure private-key functional encryption scheme for $P$. 

12
Proof Sketch. The proof proceeds by a sequence of hybrids. For simplicity, we consider the case when the adversary submits a single challenge pair \((m_0, m_1)\), and the argument can be easily generalized to the case of multiple challenges.

**Hybrid_0:** This corresponds to the real experiment where the challenger sends an encryption of \(m_0\) to the adversary.

**Hybrid_1:** For every functional query \(F\), the challenger replaces \(C_E\) with a symmetric encryption \(\text{Sym.Enc}(K_E, \hat{F}(m_0; \text{PRF}_{K_P}(t)))\) in the functional key for \(F\). By a sequence of intermediate hybrids (as many as the number of function queries), \(\text{Hybrid}_1\) can be shown to be computationally indistinguishable from \(\text{Hybrid}_0\) based on the pseudorandom ciphertexts property of the symmetric encryption scheme.

**Hybrid_2:** The challenge ciphertext will consist of an encryption of \((m_0, 0, K_E, 1)\) instead of \((m_0, K_P, 0^k, 0)\). This hybrid is computationally indistinguishable from \(\text{Hybrid}_1\) by the security of the underlying functional encryption scheme.

**Hybrid_3:** For every function query \(F\), the challenger replaces \(C_E\) in all the functional keys with \(\text{Sym.Enc}(K_E, \hat{F}(m_0; r))\) for a uniform \(r\). By a sequence of intermediate hybrids (as many as the number of function queries), \(\text{Hybrid}_3\) can be shown to be computationally indistinguishable from \(\text{Hybrid}_2\) based on the security of PRF.

**Hybrid_4:** Finally, for every function query \(F\), the challenger replaces \(\hat{F}(m_0; r)\) in the ciphertext hardwired in the functional key for \(F\) by the simulated randomized encoding \(\text{Sim}(1^k, F(m_0))\). By a sequence of intermediate hybrids (as many as the number of function queries), \(\text{Hybrid}_4\) can be shown to be computationally indistinguishable from \(\text{Hybrid}_3\) based on the security of randomized encodings. Note that the this hybrid does not depend on whether \(m_0\) or \(m_1\) was encrypted since for all function queries \(F\) it holds that \(F(m_0) = F(m_1)\), and this proves the security of \(\mathcal{PFE}\).

References


A Preliminaries (Cont.)

A.1 Private-Key Functional Encryption

A private-key functional encryption (FE) scheme \( \text{Priv} \) over a message space \( \mathcal{M} = \{ \mathcal{M}_\lambda \}_{\lambda \in \mathbb{N}} \) and a function space \( \mathcal{F} = \{ \mathcal{F}_\lambda \}_{\lambda \in \mathbb{N}} \) is a tuple \( (\text{Priv.Setup}, \text{Priv.KeyGen}, \text{Priv.Enc}, \text{Priv.Dec}) \) of PPT algorithms with the following properties:

- \( \text{Priv.Setup}(1^\lambda) \): The setup algorithm takes as input the unary representation of the security parameter, and outputs a secret key \( \text{Priv.MSK} \).
- \( \text{Priv.KeyGen}(\text{Priv.MSK}, f) \): The key-generation algorithm takes as input the secret key \( \text{Priv.MSK} \) and a function \( f \in \mathcal{F}_\lambda \), and outputs a functional key \( \text{Priv.sk}_f \).
- \( \text{Priv.Enc}(\text{Priv.MSK}, m) \): The encryption algorithm takes as input the secret key \( \text{Priv.MSK} \) and a message \( m \in \mathcal{M}_\lambda \), and outputs a ciphertext \( \text{CT} \).
- \( \text{Priv.Dec}(\text{Priv.sk}_f, \text{CT}) \): The decryption algorithm takes as input a functional key \( \text{Priv.sk}_f \) and a ciphertext \( \text{CT} \), and outputs \( m \in \mathcal{M}_\lambda \cup \{ \perp \} \).
In terms of correctness, we require that there exists a negligible function $\operatorname{negl}(\cdot)$ such that for all sufficiently large $\lambda \in \mathbb{N}$, for every message $m \in \mathcal{M}_\lambda$, and for every function $f \in \mathcal{F}_\lambda$ it holds that

$$\operatorname{Priv.Enc}(\operatorname{Priv.KeyGen}(\operatorname{Priv.MSK}, f), \operatorname{Priv.Enc}(\operatorname{Priv.MSK}, m)) = f(m)$$

with probability at least $1 - \operatorname{negl}(\lambda)$, where $\operatorname{Priv.MSK} \leftarrow \operatorname{Priv.Setup}(1^\lambda)$, and the probability is taken over the random choices of all algorithms.

We consider the standard selective and adaptive indistinguishability-based notions for private-key functional encryption (see, for example, [BS15]). Intuitively, these notions ask that encryptions of any two messages, $m_0$ and $m_1$, should be computationally indistinguishable given access to functional keys for any function $f$ such that $f(m_0) = f(m_1)$ and to an encryption oracle.

**Definition 4 (Selective security).** A private-key functional encryption scheme $\Pi = (\text{Sel.Setup, Sel.KeyGen, Sel.Enc, Sel.Dec})$ over a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ and a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ is selectively secure if for any PPT adversary $A$ there exists a negligible function $\operatorname{negl}(\cdot)$ such that

$$\operatorname{Adv}_{\Pi, A}^{\text{Sel}}(\lambda) = \left| \Pr[\operatorname{Expt}_{\Pi, A}^{\text{Sel}}(\lambda, 0) = 1] - \Pr[\operatorname{Expt}_{\Pi, A}^{\text{Sel}}(\lambda, 1) = 1] \right| \leq \operatorname{negl}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$, where for each $b \in \{0, 1\}$ and $\lambda \in \mathbb{N}$ the experiment $\operatorname{Expt}_{\Pi, A}^{\text{Sel}}(1^\lambda, b)$, modeled as a game between the adversary $A$ and a challenger, is defined as follows:

1. **Setup phase:** The challenger samples $\text{Sel.MSK} \leftarrow \text{Sel.Setup}(1^\lambda)$.

2. **Message queries:** On input $1^\lambda$ the adversary submits $((m_1^{(0)}, \ldots, m_p^{(0)}), (m_1^{(1)}, \ldots, m_p^{(1)}))$ for some polynomial $p = p(\lambda)$. The challenger replies with $(c_1, \ldots, c_p)$, where $c_i \leftarrow \text{Sel.Enc}(\text{Sel.MSK}, m_i^{(b)})$ for every $i \in [p]$.

3. **Function queries:** The adversary adaptively queries the challenger with any function $f \in \mathcal{F}_\lambda$ such that $f(m_i^{(0)}) = f(m_i^{(1)})$ for every $i \in [p]$. For each such query, the challenger replies with $\text{Sel.sk}_f \leftarrow \text{Sel.KeyGen}(\text{Sel.MSK}, f)$.

4. **Output phase:** The adversary outputs a bit $b'$ which is defined as the output of the experiment.

**Definition 5 (Adaptive security).** A private-key functional encryption scheme $\Pi = (\text{Ad.Setup, Ad.KeyGen, Ad.Enc, Ad.Dec})$ over a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ and a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ is adaptively secure if for any PPT adversary $A$ there exists a negligible function $\operatorname{negl}(\cdot)$ such that

$$\operatorname{Adv}_{\Pi, A}^{\text{Ad}}(\lambda) = \left| \Pr[\operatorname{Expt}_{\Pi, A}^{\text{Ad}}(\lambda, 0) = 1] - \Pr[\operatorname{Expt}_{\Pi, A}^{\text{Ad}}(\lambda, 1) = 1] \right| \leq \operatorname{negl}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$, where for each $b \in \{0, 1\}$ and $\lambda \in \mathbb{N}$ the experiment $\operatorname{Expt}_{\Pi, A}^{\text{Ad}}(1^\lambda, b)$, modeled as a game between the adversary $A$ and a challenger, is defined as follows:

1. **Setup phase:** The challenger samples $\text{Ad.MSK} \leftarrow \text{Ad.Setup}(1^\lambda)$.

2. **Query phase:** The adversary adaptively queries the challenger with message queries and function queries, in an arbitrary order, as follows:

   - **Message queries:** The adversary submits $(m_0, m_1)$ such that $f(m_0) = f(m_1)$ for all function queries $f$ made so far. The challenger replies with $CT = \text{Ad.Enc(Ad.MSK, m_0)}$.

   - **Function queries:** The adversary submits a function $f$ such that $f(m_0) = f(m_1)$ for all message queries $(m_0, m_1)$ made so far. The challenger replies with $\text{Ad.sk}_f \leftarrow \text{Ad.KeyGen(Ad.MSK, f)}$.

3. **Output phase:** The adversary outputs a bit $b'$ which is defined as the output of the experiment.
B Our Transformation in the Private-Key Setting

The exact same transformation as above works in the private-key setting as well. Namely, given a private-key selectively secure FE, we obtain a private-key adaptively secure FE. The transformation is identical with the obvious exception that there is no public-key, and the master secret key is used for both encryption and key generation. We denote the selectively-secure FE that we use by Sel = (Sel.Setup, Sel.KeyGen, Sel.Enc, Sel.Dec). The adaptively-secure FE that we construct is denoted by Ad = (Ad.Setup, Ad.KeyGen, Ad.Enc, Ad.Dec).

Ad.Setup(1^\lambda): Execute Sel.Setup(1^\lambda) to obtain Sel.MSK. Output Ad.MSK = Sel.MSK.

Ad.KeyGen(Ad.MSK = Sel.MSK, f):

- Pick a uniformly random string C_E \leftarrow \{0,1\}^{f_1(\lambda)} and a uniformly random tag \tau \leftarrow \{0,1\}^{f_2(\lambda)}.
- Define the circuit
  \[ G_{f,C_E,\tau}(\text{OneCT}.SK, K, \text{Sym}.K, \beta) = \begin{cases} \text{OneCT}.sk_f \leftarrow \text{OneCT}.KeyGen(\text{OneCT}.SK, f; \text{PRF}_K(\tau)) & \text{if } \beta = 0 \\ \text{Sym}.Dec(\text{Sym}.K, C_E) & \text{if } \beta = 1 \end{cases} \]
  where \( C_E \in \{0,1\}^{f_1(\lambda)} \) and \( \tau \in \{0,1\}^{f_2(\lambda)} \) are as above. Furthermore, \( K, \text{Sym}.K \in \{0,1\}^\lambda \), and \( \beta \in \{0,1\} \).
- Run Sel.sk_G \leftarrow Sel.KeyGen(Sel.MSK, G_{f,C_E,\tau}) and outputs Ad.sk_f = Ad.sk_G.

Ad.Enc(Ad.MSK = Sel.MSK, m):

- Execute OneCT.Setup(1^\lambda) to obtain OneCT.SK.
- Sample K from the appropriate PRF key space.
- Execute OneCT.Enc(OneCT.SK, m) to obtain CT_0. Execute Sel.Enc(Sel.MSK, M = (OneCT.SK, K, 0^\lambda, 0)) to obtain CT_1.
- Output CT = (CT_0, CT_1).

Ad.Dec(Ad.sk_f = Sel.sk_G, CT = (CT_0, CT_1)): Execute Sel.Dec(Sel.sk_G, CT_1) to obtain OneCT.sk_f. Execute OneCT.Dec(OneCT.sk_f, CT_0) to obtain m.

The correctness is straightforward. The proof of security in this case is slightly more complicated than its public-key counterpart. Since in the symmetric setting, the adversary is allowed to make multiple message queries, we have to employ a sequence of hybrids, handling each message query at a time. Each of these hybrids is identical to our proof of Theorem 2 above.

Security. We show that any PPT adversary \( \mathcal{A} \) succeeds in the adaptive security game of Ad with only negligible probability. We will show this in a sequence of hybrids. We denote the advantage of the adversary in Hybrid^1 \( _i \) to be the probability that the adversary outputs 1 in that hybrid and this quantity is denoted by Adv^1_{\mathcal{A},i}.

We define the hybrids in the following manner and we prove indistinguishability of every two consecutive hybrids (two hybrids are consecutive if they are connected by an arrow). The text on top of the arrow indicates the claim we use to prove the indistinguishability. The symbol “⇒” on top of the arrow indicates that the consecutive hybrids are identical. We denote by \( p \) the number of message queries made by \( \mathcal{A} \).

\[
\begin{align*}
\text{Hybrid}^0_0 & \Rightarrow \text{Hybrid}^1_{1,0} \overset{\text{Claim}^1_1}{\Rightarrow} \text{Hybrid}^2_{1,0} \overset{\text{Claim}^2_1}{\Rightarrow} \text{Hybrid}^3_{1,0} \overset{\text{Claim}^3_1}{\Rightarrow} \text{Hybrid}^4_{1,0} \\
\cdots \text{Hybrid}^1_{4,0} & \overset{\text{Claim}^4_4}{\Rightarrow} \text{Hybrid}^1_{4,1} \overset{\text{Claim}^3_3}{\Rightarrow} \text{Hybrid}^1_{3,1} \overset{\text{Claim}^2_2}{\Rightarrow} \text{Hybrid}^1_{2,1} \overset{\text{Claim}^1_1}{\Rightarrow} \text{Hybrid}^1_{1,1}
\end{align*}
\]
Hybrid_0: This corresponds to the real experiment when the challenger uses the encryption oracle, parameterized by bit 0, to generate the challenge ciphertexts. That is, for all message queries of the form \((m_0, m_1)\), the challenger sends an encryption of \(m_0\) to the adversary. The output of this hybrid is the same as the output of the adversary.

Hybrid_{1,b} for \(b \in \{0, 1\}, j \in [p]\): This is the same as the hybrid \(\text{Hybrid}_{1, b}^{-1}\) (if \(j = 1\) then we refer to \(\text{Hybrid}_0\)) except that the challenger encrypts the \(b\)th message in the \(j\)th message pair query submitted by the adversary. More precisely, the only change is the following: If the adversary submits the \(j\)th message pair \((m_0, m_1)\) to the challenger, the challenger then sends the challenge ciphertext \(CT^*\) back to the adversary, where \(CT^*\) is the encryption of message \(m_b\).

We observe that the hybrid \(\text{Hybrid}_{1, 0}\) is identical to \(\text{Hybrid}_0\) and also, \(\text{Hybrid}_{1, 1}^{-1}\) is identical to \(\text{Hybrid}_1\), for \(j \in [p]\) and \(j > 1\).

Hybrid_{2,b} for \(b \in \{0, 1\}, j \in [p]\): This is identical to the previous hybrid, \(\text{Hybrid}_{1,b}\), except for the following change. The challenger replaces \(C_E\) in every functional key, corresponding to the query \(f\) made by the adversary, with a symmetric encryption of \(\text{OneCT}.sk_f\), where \(\text{OneCT}.sk_f\) is the output of \(\text{OneCT}.\text{KeyGen}(\text{OneCT}.SK^*, f; \text{PRF}_{K^*}(\tau))\) and \(K^*\) is a PRF key sampled from the keyspace \(K\). Further the symmetric encryption is computed with respect to \(\text{Sym}.K^*\), where \(\text{Sym}.K^*\) is the output of \(\text{Sym}.\text{Setup}(1^\lambda)\) and \(\tau\) is the tag associated to the functional key of \(f\). We emphasize that the same \(\text{Sym}.K^*\) and \(K^*\) is used while generating all the functional keys.

Claim 1. Assuming the pseudorandom ciphertext property of \(\text{SYM}\), for every PPT adversary \(A\), for \(b \in \{0, 1\}, j \in [p]\), we have \(|\text{Adv}_{A, b, j}^1 - \text{Adv}_{A, b, j}^2| \leq \text{neg}(\lambda)|.

Proof. Suppose there exists an adversary such that the difference in the advantages is non-negligible, then we construct a reduction that can break the security of \(\text{SYM}\). The reduction internally executes the adversary by simulating the role of the challenger in the adaptive private-key FE game. It answers both the message and the functional queries made by the adversary as follows. The reduction first executes \(\text{OneCT}.\text{Setup}(1^\lambda)\) to obtain \(\text{OneCT}.SK^*\). It then samples \(K^*\) from \(K\). Further, the reduction generates \(\text{Sel}.\text{MSK}\), which is the output of \(\text{Sel}.\text{Setup}(1^\lambda)\) and \(\text{Sym}.K^*\), which is the output of \(\text{Sym}.\text{Setup}(1^\lambda)\). When the adversary submits a functional query \(f\), the reduction first picks \(\tau\) at random. The reduction executes \(\text{OneCT}.\text{KeyGen}(\text{OneCT}.SK^*, f; \text{PRF}(K^*(\tau)))\) to obtain \(\text{OneCT}.sk_f\). It then sends \(\text{OneCT}.sk_f\) to the challenger of the symmetric encryption scheme. The challenger returns back with \(C_E\), where \(C_E\) is either a uniformly random string or it is an encryption of \(\text{OneCT}.sk_f\). The reduction then generates a selectively-secure FE functional key of \(G_{f, C_E, \tau}\) and denote the result by \(\text{Sel}.sk_G\) which is sent to the adversary. The message queries made by the adversary are handled as in \(\text{Hybrid}_{1, b}\). That is, the adversary submits the \(i\)th message-pair query of the form \((m_0, m_1)\) and the reduction sends \(CT^* = (CT^*_0, CT^*_1)\) back to the adversary, where \(CT^*_0 = \text{OneCT}.\text{Enc}(\text{OneCT}.SK^*, m_b)\) and \(CT^*_1 = \text{Sel}.\text{Enc}(\text{Sel}.\text{MSK}, (0^\lambda, 0^\lambda, \text{Sym}.K^*, 1))\); we define \(b_i = 1\) for \(i < j, b_j = b\) and for \(i > j\), we have \(b_i = 0\).

If the challenger of the symmetric key encryption scheme sends a uniformly random string back to the reduction every time the reduction makes a query to the challenger then we are in \(\text{Hybrid}_{1, b}\), otherwise we are in \(\text{Hybrid}_{2, b}\). Since the adversary can distinguish both the hybrids with non-negligible probability, we have that the reduction breaks the security of the symmetric key encryption scheme with non-negligible probability. This proves the claim.

Hybrid_{2,b} for \(b \in \{0, 1\}, j \in [p]\): The challenger modifies the \(j\)th challenge ciphertext \(CT^* = (CT^*_0, CT^*_1)\). In particular it generates \(CT^*_1\) using the message \((0^\lambda, 0^\lambda, \text{Sym}.K^*, 1)\) instead of \((\text{OneCT}.SK^*, K^*, 0^\lambda, 0)\). The ciphertext component \(CT^*_0\) is generated the same way as in the previous hybrid, \(\text{Hybrid}_{2, b}\).

More formally, the \(j\)th challenge ciphertext is now \(CT^* = (CT^*_0 = \text{OneCT}.\text{Enc}(\text{OneCT}.SK^*, m_b), CT^*_1 = \text{Sel}.\text{Enc}(\text{Sel}.\text{MSK}, (0^\lambda, 0^\lambda, \text{Sym}.K^*, 1)))\). The rest of the hybrid is the same as the previous hybrid, \(\text{Hybrid}_{2, b}\).
Claim 2. Assuming the selective security of Sel, for every PPT adversary $A$, for $b \in \{0,1\}$, $j \in [p]$, we have $|Adv_{A,b,j}^2 - Adv_{A,b,j}^3| \leq \text{negl}(\lambda)$.

Proof. Suppose the claim is not true for some PPT adversary $A$, we construct a reduction that breaks the security of Sel. Our reduction will internally execute $A$ by simulating the role of the challenger of the adaptive FE game.

For every $i \in [p]$, the reduction first executes OneCT.Setup(1$^\lambda$) to obtain OneCT.SK$^*_i$. It then samples $K^*_i$ from $K$. It also executes Sym.Setup(1$^\lambda$) to obtain Sym.K$^*_i$. If $i \neq j$, the reduction then sends the message pair $((\text{OneCT.SK}^*_j, K^*_j, 0^\lambda, 0))$, $(\text{OneCT.SK}^*_i, K^*_i, 0^\lambda, 0)$ to the challenger of the selective game and if $i = j$, the reduction instead sends the message pair $((\text{OneCT.SK}^*_j, K^*_j, 0^\lambda, 0), (0^\lambda, 0^\lambda, \text{Sym.K}^*_i, 1))$. For the $i^{th}$ message query, the challenger responds back with the challenge ciphertext $\text{CT}_i$.

The reduction is now ready to interact with the adversary $A$. If $A$ makes a functional query $f$ then the reduction constructs the circuit $G_{f,C_{E,\tau}}$ as in Hybrid$^2_{4,b}$. It then queries the challenger of the selective game with the function $G$ and in return it gets the key $\text{Sel.sk}_G$. The reduction then sets $\text{Ad.sk}_f$ to be $\text{Sel.sk}_G$ which it then sends back to $A$. The message queries made by $A$ are handled as follows. When $A$ submits the $i^{th}$ message query $(m_0, m_1)$, the reduction executes OneCT.Enc($\text{OneCT.SK}^*_i, m_0$) to obtain $\text{CT}^*_i$. It then sends the ciphertext $\text{CT}^*_i = (\text{CT}^*_0, \text{CT}^*_1, \ldots)$ to the adversary. The output of the reduction is the output of $A$.

We claim that the reduction is a legal adversary in the selective security game of Sel. To argue this, note that we only need to consider the $j^{th}$ message query since the left and right messages in all other message queries are the same. For the $j^{th}$ message query $(M_0) = (\text{OneCT.SK}^*_j, K^*_j, 0^\lambda, 0)$, $(M_1) = (0^\lambda, 0^\lambda, \text{Sym.K}^*_j, 1)$ and every functional query of the form $G_{f,C_{E,\tau}}$ made by the reduction, we have that $G_{f,C_{E,\tau}}(M_0) = G_{f,C_{E,\tau}}(M_1)$: By definition, $G_{f,C_{E,\tau}}(M_0)$ is the functional key of $f$, with respect to key OneCT.SK$^*_j$ and randomness PRF$^i(K^*_j(\tau))$. Further, $G_{f,C_{E,\tau}}(M_1)$ is the encryption of $C_E$ which is nothing but the functional key of $f$, with respect to key OneCT.SK$^*_j$ and randomness PRF$^i(K^*_j(\tau))$. This proves that the reduction is a legal adversary in the selective security game.

If the challenger of the selective game sends back an encryption of $(\text{OneCT.SK}^*_j, K^*_j, 0^\lambda, 0)$ then we are in Hybrid$^2_{4,b}$ else if the challenger encrypts $(0^\lambda, 0^\lambda, \text{Sym.K}^*_j, 1)$ then we are in Hybrid$^3_{4,b}$. By our hypothesis, this means the reduction breaks the security of the selective game with non-negligible probability that contradicts the security of Sel. This completes the proof of the claim.

Hybrid$^4_{4,b}$ for $b \in \{0,1\}$, $j \in [p]$: For every functional query $f$ made by the adversary, the challenger generates $C_E$ by executing Sym.Enc(Sym.K$^*_j$, OneCT.sk$^*_f$), with OneCT.sk$^*_f$ being the output of OneCT.KeyGen(OneCT.SK$^*_j$, $f$; $R$), where $R$ is picked at random. The rest of the hybrid is the same as the previous hybrid.

Claim 3. Assuming the security of the pseudorandom function family $F$, for every PPT adversary $A$, for $b \in \{0,1\}$, $j \in [p]$, we have $|Adv_{A,b,j}^4 - Adv_{A,b,j}^3| \leq \text{negl}(\lambda)$.

Proof. Suppose the claim is false for some PPT adversary $A$, we construct a reduction that internally executes $A$ and breaks the security of the pseudorandom function family $F$. The reduction simulates the role of the challenger of the adaptive game when interacting with $A$. The reduction answers the functional queries, made by the adversary as follows; the message queries are answered as in Hybrid$^4_{3,b}$ (or Hybrid$^4_{4,b}$). For every functional query $f$ made by the adversary, the reduction picks $\tau$ at random which is then forwarded to the challenger of the PRF security game. In response it receives $R^\tau$. The reduction then computes $C_E$ to be Sym.Enc(Sym.K$^*_j$, OneCT.sk$^*_f$), where OneCT.sk$^*_f$ = OneCT.KeyGen(OneCT.SK$^*_j$, $f$; $R^\tau$). The reduction then proceeds as in the previous hybrids to compute the functional key Ad.sk$^*_f$ which it then sends to $A$.

If the challenger of the PRF game sent $R^\tau = \text{PRF}^i(K(\tau))$ back to the reduction then we are in Hybrid$^4_{3,b}$ else if $R^\tau$ is generated at random, for every query $\tau$, by the challenger then we are in Hybrid$^4_{4,b}$. From our hypothesis this means that the probability that the reduction distinguishes the pseudorandom values from random values is non-negligible, contradicting the security of the pseudorandom function family $F$.

We now show that Hybrid$^4_{4,0}$ is computationally indistinguishable from Hybrid$^4_{4,1}$. 

\[19\]
Claim 4. Assuming the adaptive security of OneCT, for $j \in [p]$, for every PPT adversary $A$ we have $|\text{Adv}_{4,j}^A - \text{Adv}_{4,1,j}^A| \leq \text{negl}(\lambda)$.

Proof. Suppose there exists a PPT adversary $A$, such that the claim is false. We design a reduction that internally executes $A$ to break the adaptive security of OneCT.

The reduction simulates the role of the challenger of the adaptive private-key FE game. It answers both the functional as well as message queries made by the adversary as follows. If $A$ makes a functional query $f$ then it forwards it to the challenger of the adaptively-secure single-ciphertext FE scheme. In return it receives $\text{OneCT}.sk_f$. It then encrypts it using the symmetric encryption scheme, where the symmetric key is picked by the reduction itself, and denote the resulting ciphertext to be $C_E$. The reduction then constructs the circuit $G_{f,C_E,T}$ as in the previous hybrids. Finally, the reduction computes the selective private-key functional key of $G_{f,C_E,T}$, where the reduction itself picks the master secret key of selective private-key FE scheme. The resulting functional key is then sent to $A$. The message queries are handled as follows. Suppose the adversary $A$ makes the $i^{th}$ message-pair query $(m_0^i, m_1^i)$. If $i \neq j$, then the reduction answers the query himself. That is, $B$ samples the single-ciphertext FE master key $\text{OneCT}.SK_i$ and PRF key $K_i$ by himself. It then computes a single-ciphertext FE encryption of $m_i$ using $\text{OneCT}.SK_i$ and denote the result by $CT_0^i$: we define $b_i = 1$ if $i < j$ and $b_i = 0$ if $i > j$. Further, it computes a (selective) private-key FE encryption of $(\text{OneCT}.SK_i, K_i, 0^\lambda, 0)$, which is represented by $CT_1^i$. The challenger sends the ciphertext $CT_i = (CT_0^i, CT_1^i)$ to $A$. When $i = j$, the reduction forwards the message pair $(m_0^i, m_1^i)$ to the challenger of the adaptive game. In response it receives $CT_0^*$. The reduction then generates $CT_1^*$ on its own where $CT_1^*$ is the selective FE encryption of $(0^\lambda, 0^\lambda, \text{Sym}.K^*, 1)$. The reduction then sends $CT^* = (CT_0^*, CT_1^*)$ to $A$. The output of the reduction is the output of $A$.

We note that the reduction is a legal adversary in the adaptive game of OneCT, i.e., for the message query $(m_0^i, m_1^i)$, functional query $f$, we have that $f(m_0^i) = f(m_1^i)$: this follows from the fact that (i) the functional queries (resp., challenge message query) made by the adversary (of Ad) is the same as the functional queries (resp., challenge message query) made by the reduction, and (ii) the adversary (of Ad) is a legal adversary. This proves that the reduction is a legal adversary in the adaptive game.

If the challenger sends an encryption of $m_0$ then we are in $\text{Hybrid}_{4,0}^j$ and if the challenger sends an encryption of $m_1$ then we are in $\text{Hybrid}_{4,1}^j$. From our hypothesis, this means that the reduction breaks the security of OneCT. This proves the claim. \hfill $\Box$

$\text{Hybrid}_5$: This corresponds to the real experiment when the challenger uses the encryption oracle, parameterized by bit 1, to generate the challenge ciphertexts. That is, for all message queries of the form $(m_0, m_1)$, the challenger sends an encryption of $m_1$ to the adversary. The output of this hybrid is the same as the output of the adversary.

We note that this hybrid is identical to the hybrid $\text{Hybrid}_{4,1}^0$.

The above claims imply that $\text{Hybrid}_0$ is computationally indistinguishable from $\text{Hybrid}_5$ which proves the adaptive security of $\text{Ad}$. We thus have the following theorem.

Theorem 4. Assuming the existence of a sufficiently-expressive selectively-secure private-key functional encryption scheme, there exists an adaptively-secure private-key functional encryption scheme.