A Definitional Framework for Functional Encryption*

Christian Matt and Ueli Maurer
Department of Computer Science
ETH Zurich, Switzerland
email: {mattc, maurer}@inf.ethz.ch

Abstract

Functional encryption (FE) is a powerful generalization of various types of encryption. We investigate how FE can be used by a trusted authority to enforce access-control policies to data stored in an untrusted repository. Intuitively, if (functionally) encrypted data items are put in a publicly-readable repository, the effect of the encryption should be that every user has access to exactly (and only) those functions of the data items for which he has previously received the corresponding decryption key. That is, in an ideal-world view, the key authority can flexibly manage read access of users to the repository. This appears to be exactly what FE is supposed to achieve, and most natural applications of FE can be understood as specific uses of such a repository with access control. However, quite surprisingly, it is unclear whether known security definitions actually achieve this goal and hence whether known FE schemes can be used in such an application. In fact, there seems to be agreement in the cryptographic community that identifying the right security definitions for FE remains open.

To resolve this problem, we treat FE in the constructive cryptography framework and propose a new conventional security definition, called *composable functional encryption security* (CFE-security), which exactly matches the described ideal-world interpretation. This definition (and hence the described application) is shown to be unachievable in the standard model but achievable in the random oracle model. Moreover, somewhat weaker definitions, which are achievable in the standard model, can be obtained by certain operational restrictions of the ideal-world repository, making explicit how schemes satisfying such a definition can (and cannot) meaningfully be used. Finally, adequate security definitions for generalizations of FE (such as multi-input, randomized functions, malicious ciphertext generation, etc.) can be obtained by straight-forward operational extensions of the repository and extracting the corresponding security definitions. This leads towards a unified treatment of the security of FE.

1 Introduction

1.1 Functional Encryption

Functional encryption (FE) is a very general concept formally introduced by Boneh, Sahai, and Waters [BSW11]. Many types of encryption such as public-key encryption, identity-based encryption [Sha85, MY96, BF01], and attribute-based encryption [SW05] can be seen as a special

*An earlier version of this paper was titled "A Constructive Approach to Functional Encryption". This is the full version of the paper that appears in the proceedings of the 28th IEEE Computer Security Foundations Symposium (CSF), 2015.

case of FE. Briefly, an FE scheme is parametrized by a value space and a set of functions on the value space. A trusted authority holding a master secret key corresponding to the master public key can generate secret keys for all functions in this set. Given the master public key one can encrypt messages, and given a secret key for a particular function f and an encryption of a value x, one can efficiently compute $f(x)$ but does not learn anything more about x. Moreover, even if one pools the secret keys for many functions $f_1, \dots f_k$, one can compute nothing about an encrypted value x except for exactly these function values, i.e., $f_1(x), \dots, f_k(x)$.

Formalizing these intuitive security requirements has caused more trouble than one might expect, and several security definitions for functional encryption exist in the literature. While some of them were shown to be too weak since schemes that should not be considered secure could be proven to satisfy them, others are so strong that even for very simple sets of functions, no scheme exists that satisfies them in the plain model [O'N10, BSW11, AGVW13, BO13, CIJ+13]. The research community seems to agree that it is unclear which definition is adequate or can even be considered the "right" one for a certain application context. This raises the question of which definition is suitable for being used in a certain application.

In an ideal-world view, however, it seems quite natural to state what one would expect from FE. If a publicly readable data repository (e.g., a public web repository) is available, then, by virtue of encrypting the data items before putting them into the repository, one constructs a repository where the data is now secret and where a specific entity has the capability of assigning to entities the right to access certain functions (but not more) of the data items stored in the repository. Obviously, granting the access right to an entity in this ideal world view corresponds in the real world to providing the corresponding function's decryption key, and accessing the data corresponds to decrypting the corresponding ciphertext.

Compared to FE, a standard public-key encryption scheme achieves the same goal of providing access control for a repository, but access would be all-or-nothing: If one knows the secret key, then one can decrypt, otherwise one cannot. A more versatile access control mechanism is achieved by identity-based encryption, which allows the trusted party to grant users the right to access data for a specific identity, where the input data contains the data itself and the identity that should be able to access it. More advanced access policies can be implemented using attribute-based encryption. The high flexibility of FE is demonstrated by the following application of FE proposed in [BSW12, GKP+13]: Assume some user receives encrypted emails that are stored on his provider's server and he does not want to download mails with a high probability of being spam, to avoid unnecessary traffic. The trusted party, which in this case can coincide with the recipient of the mails, generates a special secret key for the provider that only allows to compute the score function of the spam filter. The provider now cannot read the contents of the mails but is still able to filter out spam and notify the recipient only about the remaining incoming messages. The repository here corresponds to the mail server and inputting data into the repository to sending the recipient encrypted emails.

1.2 Problem Statement

This paper addresses the following questions: Does the above-described intuition of a secure repository really hold? Does one of the existing security definitions imply that such a view is valid, and if so, which one? What, then, do the other definitions achieve? Could it be that if a scheme satisfying an existing security definition is used, then the above straightforward

applications become insecure?[1]

1.3 Security Definitions and Their Semantics

Before explaining our approach to address these questions, we discuss a general issue with cryptographic security definitions. Security definitions for cryptographic primitives such as one-way functions, message authentication codes, public-key encryption, or FE, can serve two entirely different purposes, which are often not clearly distinguished.

The first purpose of a security definition is to serve as a (technical) reference point, on the one hand for devising schemes provably satisfying the definition based on a weak assumption (e.g., a CCA-secure PKE scheme based on the DDH assumption [CS98]), and on the other hand for building more sophisticated primitives from any scheme satisfying the definition (e.g., constructing a CCA-secure PKE scheme from a CPA-secure IBE scheme and a one-time signature scheme [BCHK06]). Results about security definitions often take the form of a comparison, for example an equivalence or a separation statement, meaning that one definition is strictly stronger than another one.

The second purpose of a (technical) security definition is to assure the security of a certain type of application when a scheme satisfying the (technical) security definition is used. While definitions are usually devised with much intuition for what is needed in a certain application (and indeed the definition is often motivated by an application story), it is important to point out that a conventional technical security definition for a cryptographic primitive can generally not directly imply the security of an associated application, for two independent reasons. First, the particular *use* of the primitive within a protocol would have to be precisely specified. For example how is the message to be encrypted formed, to whom is it sent, over what kind of channel, and are certain fields like an IP address included in the MAC? Second, the application and its security requirements would also have to be formalized precisely. For example, game-based security definitions for key agreement do not explicitly guarantee that one can safely use the key in a given context, nor what the requirements are for the channels over which the protocol is executed (e.g., that they must be authenticated).

1.4 Addressing the Problem

The general question of bridging the gap between a technical (e.g., game-based) security definition and the security of an intended application making use of the primitive is a foundational one, and it applies also to other cryptographic primitives. Exploring it seems particularly valuable for FE because, unlike for other cryptographic primitives, definitional issues do not seem to be settled.

One goal of the constructive cryptography (CC) framework [MR11, Mau12] is to do exactly this: provide constructive semantics for technical security definitions and, based on the semantics, to compare definitions and identify the adequate one(s).

[1] A striking example of a failure of a security definition, which is not apparent when examining the definition and went unnoticed for almost 20 years, was demonstrated in [KRBM07] where a provably-secure quantum key agreement protocol was shown which becomes insecure when the key is used in the one-time pad encryption (the application). How can such a mismatch happen, and how can it be prevented? The definition was simply not strong enough for a generated key to be usable in an application. (This was corrected in [Ren05], where the now established, substantially stronger security definition was stated.) In a nutshell, the security definition for key agreement must guarantee that the generated key is as good as a uniform key, in any application.

The approach taken is well-established in cryptography and very natural, and it can perhaps be seen as the only viable approach to tightly link security definitions and applications. Namely, one formalizes the application as an ideal-world system, called a resource in CC, which captures both what one wants and what one does not want to happen. For example, secure communication can be modeled as a secure channel where the adversary learns at most the length of the message. Other frameworks that capture security properties by defining an ideal functionality include (variants of) Universal Composability (UC) [Can01, HS13, KT13] and Reactive Simulatability (RSIM) [PW01, BPW07].[2] These frameworks have a different focus and are designed bottom-up from a specific machine model, while the constructive cryptography framework follows a top-down approach, leading to simpler descriptions and avoiding technicalities.

The use of a cryptographic scheme can be understood as constructing a certain resource from certain assumed resources. For example, symmetric encryption can then be understood as constructing a secure (length-leaking) channel from an authenticated channel and a secret key, and a key-agreement protocol can be understood as constructing a secret key (distributed to two parties A and B, where the adversary learns nothing) from a bidirectional authenticated channel. If the construction notion is defined correctly, this constructive approach provides composition of constructions, i.e., the constructed resource (e.g., a secret key) can be used in any other construction as an assumed resource, and we have modularity since the overall security follows automatically from the individual security proofs.

1.5 Contributions of this Paper

We show that the exact characterization of FE as the construction of a certain access-controlled repository from a public repository and certain channels (to transmit secret keys and public keys) is indeed formally correct. This means, in particular, that one can compose constructions, according to the composition theorem of constructive cryptography. Concretely, if one has designed an application by defining it "on top" of a certain (assumed) repository with access control, then this application remains secure if the repository is implemented using a public repository, where data is encrypted using an FE scheme satisfying the appropriate definition.

However, for this to be true, none of the existing security definitions seem to suffice. Therefore we propose a new conventional security definition for FE, called CFE-security, derived from the constructive viewpoint, which corresponds to an adequately modified version of an established game-based definition by Boneh, Sahai, and Waters [BSW11, Definition 4]. This suggests that CFE-security is the appropriate definition if strong guarantees are required. We show that, as the definition in [BSW11], CFE-security is impossible to achieve in the standard model but achievable in the random oracle model. In doing so, we also exemplify how results in the (programmable) random oracle model can be translated to a construction in CC.

The adequacy of CFE-security might seem surprising since it is similar to [BSW11, Definition 4] and an example from [BF13] was supposed to show that this definition is insufficient. We recall that example and explain why the criticism is invalid. This demonstrates that traditional security definitions for FE are not well understood, which leads to misconceptions that do not arise in the constructive approach.

Moreover, we show that a weaker security definition by Gorbunov, Vaikuntanathan, and Wee [GVW12, Definition 1], which *is* achievable in the standard model, is sufficient for constructing a

[2] The paper [SPPM13] provides an ideal functionality for functional encryption in UC, but only for a special class of functional encryption schemes, namely attribute-based encryption. Thus, their results are not suitable to derive a general definitional framework for functional encryption as we do in this paper.

repository that restricts the number and order of interactions, making explicit how such schemes can (and cannot) meaningfully be used.

Finally, we show how adequate security definitions for generalizations of FE (multi-input, randomized functions, malicious ciphertext generation, etc.) can be obtained by straight-forward extensions of the repository. For a constructive security definition (requiring that a certain repository be constructed), one can extract the corresponding conventional security definition and compare it to existing generalized definitions. We conjecture that the latter do generally not correspond to a meaningful construction of a repository, but carrying out the complete analysis for all definitions is beyond the scope of this paper.

Overall, this leads to a unified treatment of many FE variants and makes explicit how they can be composed within a higher-level protocol.

2 Preliminaries

2.1 Resources, Converters, and Distinguishers

The results in this paper are formulated using the theory of constructive cryptography. In this section, we introduce the relevant concepts, following [MR11] and the exposition given in [MRT12]. We consider different types of *systems*, which are objects with *interfaces* via which they interact with their environment. Interfaces are denoted by uppercase letters. One can compose two systems by connecting one interface of each system. The composed object is again a system.

The types of systems we consider are *resources*, *converters*, and *distinguishers*. Resources have a finite set $\mathcal{I}$ of interfaces and are denoted by upper-case bold-face letters or upper-case sans-serif fonts. Resources with interface set $\mathcal{I}$ are called *$\mathcal{I}$-resources*. Converters have one *inner* and one *outer interface* and are denoted by lowercase Greek letters. The inner interface of a converter α can be connected to interface $I \in \mathcal{I}$ of a resource $\mathbf{R}$, which yields a new resource denoted by $\alpha^I\mathbf{R}$. The outer interface of α then serves as the new interface I of the composed resource $\alpha^I\mathbf{R}$. We also write $\alpha_I\mathbf{R}$ instead of $\alpha_I^I\mathbf{R}$ for a converter α_I. The sequential composition of two converters α and β is naturally defined by $(\alpha \circ \beta)^I\mathbf{R} \coloneqq \alpha^I(\beta^I\mathbf{R})$. For $\mathcal{I}$-resources $\mathbf{R}_1, \ldots, \mathbf{R}_m$, the *parallel composition* $[\mathbf{R}_1, \ldots, \mathbf{R}_m]$ is defined as the $\mathcal{I}$-resource where each interface $I \in \mathcal{I}$ allows to access the corresponding interfaces of all sub-systems $\mathbf{R}_i$ as sub-interfaces. The parallel composition $[\alpha_1, \ldots, \alpha_m]$ of converters $\alpha_1, \ldots, \alpha_m$ is defined by $[\alpha_1, \ldots, \alpha_m]^I[\mathbf{R}_1, \ldots, \mathbf{R}_m] \coloneqq [\alpha_1^I\mathbf{R}_1, \ldots, \alpha_m^I\mathbf{R}_m]$.

A *distinguisher* $\mathbf{D}$ for resources with n interfaces is a system with $n+1$ interfaces, where n of them connect to the interfaces of a resource and a bit is output at the remaining one. We write $\mathrm{P}(\mathbf{DR} = 1)$ to denote the probability that $\mathbf{D}$ outputs the bit 1 when connected to the resource $\mathbf{R}$. The goal of a distinguisher is to distinguish two resources by outputting a different bit when connected to a different resource. Its success is measured by the distinguishing advantage.

Definition 2.1. The *distinguishing advantage* of a distinguisher $\mathbf{D}$ for resources $\mathbf{R}$ and $\mathbf{S}$ is defined as

$$\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) \coloneqq \left|\mathrm{P}(\mathbf{DR} = 1) - \mathrm{P}(\mathbf{DS} = 1)\right|.$$

If $\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) = 0$ for all distinguishers $\mathbf{D}$, we say $\mathbf{R}$ and $\mathbf{S}$ are *equivalent*, denoted by $\mathbf{R} \equiv \mathbf{S}$. If

the distinguishing advantage is negligible for all efficient distinguishers, we say $\mathbf{R}$ and $\mathbf{S}$ are *computationally indistinguishable*[3], denoted by $\mathbf{R} \approx \mathbf{S}$.

2.2 Examples of Resources

An important example of resources are communication channels. They allow the sender A to send messages from the message space $M := \{0,1\}^*$ to the receiver B. We define two such channels, which differ in what an eavesdropper E learns about the messages. Outputs at interface E correspond to the information that is leaked to all dishonest parties. The channels we consider in this paper can transmit an arbitrary number of messages.

Definition 2.2. An *authenticated channel from A to B*, denoted by $\mathsf{AUT}_{A,B}$, is a resource with three interfaces A, B, and E. On input a message $m \in M$ at interface A, the same message is output at interfaces B and E.

This channel is called *authenticated* because E cannot modify the messages. The *secure channel* defined below additionally guarantees that an eavesdropper can only learn the length of the transferred messages.

Definition 2.3. A *secure channel from A to B*, denoted by $\mathsf{SEC}_{A,B}$, is a resource with three interfaces A, B, and E. On input a message $m \in M$ at interface A, the same message is output at interface B and the length $|m|$ of the message is output at interface E.

2.3 Construction of Resources

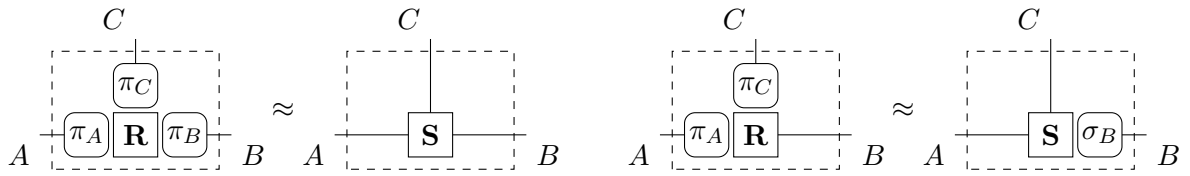
A *protocol* is a tuple of converters with the purpose of constructing a resource (with desired properties) from an assumed resource (that is available). Depending on which parties are considered potentially dishonest, we get a different notion of construction.

As an example from [CMT13], consider the setting for public-key encryption with honest A and B where we want to construct a secure channel $\mathsf{SEC}_{A,B}$ from authenticated channels $\mathsf{AUT}_{B,A}$ and $\mathsf{AUT}_{A,B}$ in presence of a dishonest eavesdropper E. Here, the assumed resource is $\mathbf{R} := [\mathsf{AUT}_{B,A}, \mathsf{AUT}_{A,B}]$ and the constructed resource is $\mathbf{S} := \mathsf{SEC}_{A,B}$. In such a setting, a protocol $\pi = (\pi_A, \pi_B)$ constructs $\mathbf{S}$ from $\mathbf{R}$ with potentially dishonest E if there exists an efficient converter σ_E (called *simulator*) such that

$$\pi_A \pi_B \bot^E \mathbf{R} \approx \bot^E \mathbf{S} \qquad \text{and} \qquad \pi_A \pi_B \mathbf{R} \approx \sigma_E \mathbf{S},$$

where $\bot$ blocks all interactions at the corresponding interface and σ_E provides a sub-interface to the distinguisher for both channels that constitute the assumed resource. The first condition is referred to as correctness condition and ensures that the protocol implements the required functionality if there is no eavesdropper. The second condition is referred to as security condition and ensures that whatever Eve can do when connected to the assumed resource without necessarily following the protocol, she could do as well when connected to the constructed resource by using the simulator σ_E.

[3] In fact, the systems considered in this paper are asymptotic objects, i.e., they correspond to families of systems indexed by a security parameter. The distinguishing advantage is a function of this parameter and efficient means that the running time is polynomial in the security parameter. To simplify the presentation, security parameters are omitted in this work.



(a) Correctness condition. (b) Security condition.

Figure 1: Depiction of the two conditions of the construction notion in Definition 2.4.

While Eve in the above example can be seen as an attacker to whom no guarantees need to be given, the setting in this paper includes one potentially dishonest party that can also be honest. Hence, all parties are assigned a protocol and all interactions provided by the constructed resource are guaranteed to all honest parties following their protocol. On the other hand, a dishonest party should not be able to do more than specified by the constructed resource. We consider a construction notion for three parties A, B, and C, where B is potentially dishonest. The following definition is a special case of the abstraction notion from [MR11] that considers many dishonest and mutually distrusting parties.

Definition 2.4. Let $\mathbf{R}$ and $\mathbf{S}$ be $\{A, B, C\}$-resources and let $\pi = (\pi_A, \pi_B, \pi_C)$ be a protocol. We say π *constructs* $\mathbf{S}$ *from* $\mathbf{R}$ *(with potentially dishonest B)*, denoted by

$$\mathbf{R} \overset{\pi}{\Longrightarrow} \mathbf{S},$$

if there exists an efficient converter σ_B such that[4]

$$\pi_A\pi_B\pi_C\mathbf{R} \approx \mathbf{S} \qquad \text{and} \qquad \pi_A\pi_C\mathbf{R} \approx \sigma_B\mathbf{S}.$$

See Figure 1 for a graphical representation of the conditions in the above definition. Similarly to the public-key encryption example above, the first condition ensures correctness of the protocol and the second condition ensures that all attacks on the protocol by B can be translated via σ_B to attacks on the constructed resource $\mathbf{S}$.

The notion of construction is composable, which intuitively means that $\mathbf{S}$ can be replaced by $\pi_A\pi_B\pi_C\mathbf{R}$ in any context without affecting the security if we have $\mathbf{R} \overset{\pi}{\Longrightarrow} \mathbf{S}$. More precisely, we have for all resources $\mathbf{R}$, $\mathbf{R}'$ $\mathbf{S}$, $\mathbf{S}'$, and $\mathbf{T}$ and for all protocols $\pi = (\pi_A, \pi_B, \pi_C)$, $\pi' = (\pi'_A, \pi'_B, \pi'_C)$, and $\phi = (\phi_A, \phi_B, \phi_C)$

$$\begin{aligned} \mathbf{R} \overset{\pi}{\Longrightarrow} \mathbf{S} \;\wedge\; \mathbf{S} \overset{\phi}{\Longrightarrow} \mathbf{T} \quad &\Longrightarrow \quad \mathbf{R} \overset{\phi\circ\pi}{\Longrightarrow} \mathbf{T}, \\ \mathbf{R} \overset{\pi}{\Longrightarrow} \mathbf{S} \;\wedge\; \mathbf{R}' \overset{\pi'}{\Longrightarrow} \mathbf{S}' \quad &\Longrightarrow \quad [\mathbf{R}, \mathbf{R}'] \overset{[\pi,\pi']}{\Longrightarrow} [\mathbf{S}, \mathbf{S}'], \end{aligned}$$

where $\phi \circ \pi \coloneqq (\phi_A \circ \pi_A, \phi_B \circ \pi_B, \phi_C \circ \pi_C)$ and $[\pi, \pi'] \coloneqq ([\pi_A, \pi'_A], [\pi_B, \pi'_B], [\pi_C, \pi'_C])$. The first property guarantees that the composition of construction steps yields a secure overall construction and the second property ensures that a construction remains secure in any context, i.e., regardless of what happens in parallel. See [MR11, Mau12] for a proof of these statements and further discussions.

[4]Note that we require the existence of one simulator for all efficient distinguishers. This simulator does therefore not depend on the distinguisher and hence is *black-box*.

2.4 Functional Encryption

A functional encryption scheme is a generalized public-key encryption scheme defined for a set F of functions with common domain X. Given the public key, one can encrypt data $x \in X$ and given a secret key for a function $f \in F$, one can efficiently compute $f(x)$ from an encryption of x. The secret keys for all $f \in F$ can be generated using a so-called master secret key which is generated together with the public key. To capture which information ciphertexts leak about the encrypted data, a special leakage function $f_0 \in F$ is considered. An intuitive security requirement guarantees that given a ciphertext for some x and secret keys for $f_1, \ldots, f_n$, one should not be able to learn more about x than what can be learned from $f_0(x), \ldots, f_n(x)$. We here only define the syntax and correctness condition of a functional encryption scheme and refer to later sections for formal security definitions. The definition here only covers unary and deterministic functions. See Section 8 for a discussion of more general notions of functional encryption.

Definition 2.5. Let X be a nonempty set and F be a set of functions with domain X such that F contains a distinguished leakage function f_0. A *functional encryption scheme* for F consists of the efficient probabilistic algorithms setup, keygen, enc, and dec. The algorithm setup generates a public key pk and a master secret key mk. Given mk and some $f \in F$, keygen generates a secret key sk_f for this f where sk_{f_0} equals the empty string. Given pk and some $x \in X$, enc computes a ciphertext c such that $\mathsf{dec}(\mathsf{sk}_f, c) \neq f(x)$ with negligible probability,[5] where the probability is over the randomness of all algorithms.

Remark. Following [BSW11], we assume everyone can always evaluate f_0. This can be seen as a rather artificial requirement; if $f_0(x)$, e.g., reveals the bit length $|x|$ of x, there has to be an efficient algorithm that precisely computes $|x|$ from a ciphertext. A more natural approach would not guarantee all parties to compute f_0, but rather not exclude in the security definition that dishonest parties can do so. To formalize that something is not guaranteed but potentially possible, the constructive cryptography framework provides the concept of filtered resources (see [MR11] for more details). While all results in this paper extend to such a definition, we stick to the definition above to simplify the presentation.

3 Repositories and Access Control

3.1 Repository Resources

In this section, we introduce a repository resource that allows users to input and access data and that naturally captures how a repository works. We first define a repository with access control and then specify a public repository without access control as a special case thereof. Users can input data from a *data set* X into the repository. After inputting data, the resource returns a *handle* (e.g., a URL or a memory address) from a set H via which the data can be accessed later. This handle could be chosen by the resource, by the user who inputs data, or by both in an interactive protocol. Since the particular procedure to generate handles is irrelevant for our purposes, we will refer to a method `getHandle` that returns an element of H without describing its implementation. We only assume that the returned handles are distinct, that is, no data is overwritten.

[5] The definition in [BSW11] requires $\mathsf{dec}(\mathsf{sk}_f, c) = f(x)$ with probability 1, but we do not need this stronger condition because our construction notion does not require perfect correctness, see Definition 2.4.

Motivated by the syntax of functional encryption, we consider a set F of *access functions* containing functions with domain X and allow users to retrieve such functions of input data. Which functions a user can access depends on the rights of this user, i.e., for each $f \in F$ users can have the right to obtain $f(x)$ for previously input $x \in X$. Everyone has the right to obtain $f_0(x)$ for a special function f_0 and a trusted authority can grant users additional rights.

We consider a resource with an interface for Alice who can input data, an interface for Bob who can access data, and an interface for the trusted party Charlie who can grant rights to Bob. Alice and Bob are not necessarily single users but correspond to roles users can have. All results in this paper regard Bob as the only potentially dishonest party. In case he is dishonest, one can also think of him as a group of dishonest and colluding parties who possibly try to combine their rights to get access to a function of some data none of them alone could access. Hence, one dishonest party is sufficient to cover collusion resistance. Similarly, the resource can be used in a context with multiple honest parties inputting data.

Definition 3.1. Let X be a nonempty set and F a set of functions with domain X and $f_0 \in F$. The resource REP_F has the interfaces A, B, and C. It internally manages the set R of functions Bob is allowed to access, and a map M assigning to a handle $h \in H$ the value $M[h] \in X \cup \{\perp\}$ where $\perp \notin X$ is a special symbol. Initially, $R = \{f_0\}$ and $M[h] = \perp$ for all $h \in H$. The resource works as follows:

Interface A

Input: $x \in X$
  $h \leftarrow$ `getHandle`
  $M[h] \leftarrow x$
  output h at interface A

Interface B

Input: $(f, h) \in F \times H$
  if $f \in R$ **and** $M[h] \neq \perp$ **then**
    output $f(M[h])$ at interface B

Interface C

Input: $f \in F$
  $R \leftarrow R \cup \{f\}$
  output f at interface B

All inputs not matching the given format are ignored.[6]

Remark. Note that Bob needs to know the handles to access data. Hence, when this resource is used in a larger protocol, Alice needs to send Bob the handles over an additional channel. This is in fact very natural: If Alice uploads a document to her web server and wants Bob to download it, she needs to tell him the URL, e.g., via email. See Appendix A for an example application of the resource that demonstrates how other protocols can use the constructed repository and how to apply the composition theorem of the constructive cryptography framework.

We now define a public repository without access control, which will serve as an assumed resource in our constructions. It corresponds to a repository as defined above where f_0 is the identity function on X, i.e., everyone is allowed to access the (identity function of) stored data.

[6] We define all resources to ignore invalid inputs. Alternatively, the resources could return error messages. While this alternative might be closer to the behavior of real systems, we decided to simply ignore invalid inputs because they are not relevant for our results.

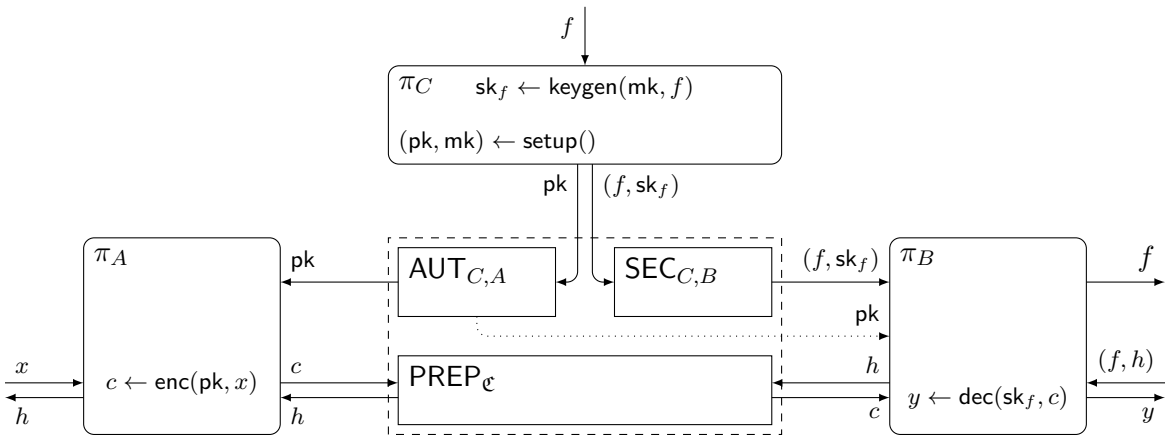


Figure 2: Overview of the protocol (π_A, π_B, π_C). The dashed rectangle represents the assumed resource, corresponding to the parallel composition of $\mathsf{AUT}_{C,A}$, $\mathsf{SEC}_{C,B}$, and $\mathsf{PREP}_{\mathfrak{C}}$, the dotted line depicts that a dishonest Bob can learn the public key while the protocol does not use it. The whole figure corresponds to a refinement of the left part of Figure 1a. By Lemma 3.3, this resource is computationally indistinguishable from REP_F.

Definition 3.2. Let X be a nonempty set, $f_0 := \mathrm{id}_X \colon X \to X, x \mapsto x$, and $P := \{f_0\}$. We define the *public repository for* X as $\mathsf{PREP}_X := \mathsf{REP}_P$. For inputs at Bob's interface, we will write h instead of (id_X, h) to simplify notation.

3.2 Access Control via Functional Encryption

A versatile repository supports a large class of access functions and restricts Bob's initial rights as much as possible. In this section, we describe how to use functional encryption to construct such a repository from a public repository. More precisely, let $\mathcal{E} = (\mathsf{setup}, \mathsf{keygen}, \mathsf{enc}, \mathsf{dec})$ be a functional encryption scheme for a set F of functions with domain X and let $\mathfrak{C}$ be the range of enc. Our goal is to construct REP_F from $\mathsf{PREP}_{\mathfrak{C}}$. To distribute keys, we additionally need an authenticated channel $\mathsf{AUT}_{C,A}$ from Charlie to Alice and a secure channel $\mathsf{SEC}_{C,B}$ from Charlie to Bob,[7] i.e., the assumed resource in our construction corresponds to $[\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}]$.

To achieve this construction, we define the protocol $\pi = (\pi_A, \pi_B, \pi_C)$ for the functional encryption scheme $\mathcal{E}$ as follows: At the beginning, π_C invokes $(\mathsf{pk}, \mathsf{mk}) \leftarrow \mathsf{setup}()$, stores mk and sends pk to Alice over the authenticated channel. This public key is internally stored by π_A. On input $x \in X$ at its outer interface, π_A outputs $c \leftarrow \mathsf{enc}(\mathsf{pk}, x)$ at its inner interface to the repository and outputs the returned handle h at its outer interface. On input $f \in F$ at its outer interface, π_C computes $\mathsf{sk}_f \leftarrow \mathsf{keygen}(\mathsf{mk}, f)$ and sends (f, sk_f) to B over the secure channel. The corresponding secret key is stored by π_B and f is output at its outer interface. On input $(f, h) \in F \times H$ at its outer interface, π_B outputs h at its inner interface to the repository if it has stored a secret key sk_f for this function f or if $f = f_0$. If it receives a ciphertext c from the repository, it outputs $y \leftarrow \mathsf{dec}(\mathsf{sk}_f, c)$ at its outer interface. All other inputs are ignored. See Figure 2 for an illustration of the protocol.

[7] For both channels, a dishonest Bob assumes the role of an eavesdropper. That is, he can learn the public key, which is sent over the authenticated channel from Charlie to Alice. If the resource is used in a context with many Bobs, it is important that the channel from Charlie to each of them is secure to prevent dishonest users from eavesdropping secret keys.

The following lemma states that this protocol constructs the desired resource if all parties are honest. It follows directly from the correctness of the functional encryption scheme.

Lemma 3.3. *For the protocol $\pi = (\pi_A, \pi_B, \pi_C)$ defined above, we have*

$$\pi_A \pi_B \pi_C[\mathsf{PREP}_{\mathfrak{E}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}] \;\approx\; \mathsf{REP}_F.$$

4 Security of Functional Encryption Schemes

4.1 Definition of CFE-Security

The protocol described in the previous section constructs the desired resource with a dishonest Bob only if the underlying functional encryption scheme satisfies a suitable security definition. We propose such a definition, based on [BSW11, Definition 4], and refer to it as *composable functional encryption security*[8] (*CFE-security* for short). We extend the definition from [BSW11] to adaptive adversaries that can choose messages depending on ciphertexts for previous messages. This extension was already mentioned in that paper but not formalized. Our definition additionally restricts oracle access of the involved algorithms. These changes are discussed after the definition. We note that SS1-security defined in [BO13] also corresponds to an adaptive variant of [BSW11, Definition 4] but also differs in other aspects and is not equivalent to the definition we propose here. In particular, SS1-security includes some auxiliary inputs that the authors claim to eliminate a weakness described in [BF13]. However, as we explain in Section 4.3, the effect pointed out in [BF13] is in fact not a weakness, so there is no need for a fix.

We follow the notation from [BSW11], i.e., for algorithms A and B, $A^{B(\cdot)}(x)$ denotes that A gets x as input and has oracle access to B, that is, $B(q)$ is answered to A in response to an oracle query q. Moreover, $A(\cdot)[[s]]$ means that A gets s as an additional input and can update its value. More precisely, $A(x)[[s]]$ corresponds to the algorithm that invokes $(y, s) \leftarrow A(x, s)$ and returns y.

Definition 4.1. Let $\mathcal{E} = (\mathsf{setup}, \mathsf{keygen}, \mathsf{enc}, \mathsf{dec})$ be a functional encryption scheme for a set F of functions with domain X. We introduce the experiments in Figure 3 for an efficient probabilistic oracle algorithm Adv_1 and efficient probabilistic algorithms Adv_2, S_1, S_2, and S_3. The advantage of a distinguisher $\mathbf{D}$ in distinguishing the outputs of these experiments is denoted by $\Delta^{\mathbf{D}}\left(\text{CFE-Exp}^{\mathrm{Real}}_{\mathcal{E},\mathrm{Adv}}, \text{CFE-Exp}^{\mathrm{Ideal}}_{\mathcal{E},\mathrm{Adv},S}\right)$.

We say $\mathcal{E}$ is *CFE-secure* if there exist S_1, S_2, and S_3 such that the distinguishing advantage is negligible for all Adv_1, Adv_2 and for all efficient distinguishers.

Note that τ can be used to share the state between Adv_1 and Adv_2 and s to share the state between S_1, S_2, and S_3. Intuitively, S_1 simulates the generation of the public key, S_2 generates simulated secret keys, and S_3 simulates ciphertexts for values x given only the images of x under the functions the adversary has already requested secret keys for. A scheme is considered secure if these simulated values are indistinguishable from the corresponding values generated by the actual protocol. The intuition is that in this case, a ciphertext does not leak anything about the encrypted value that an adversary cannot conclude from the function values he is supposed to learn.

[8] This name is justified by Theorem 4.4, which together with the composition theorem of the constructive cryptography framework implies that CFE-secure schemes indeed guarantee composability. In an earlier version of this paper, it was called "fully adaptive security" ("FA-security").

$\mathbf{CFE\text{-}Exp}^{\mathbf{Real}}_{\mathcal{E},\mathbf{Adv}}$

$(\mathsf{pk}, \mathsf{mk}) \leftarrow \mathsf{setup}()$
$(l, \tau) \leftarrow (0, 0)$
repeat
  $l \leftarrow l + 1$
  $x_l \leftarrow \mathrm{Adv}_1^{\mathsf{keygen}(\mathsf{mk},\cdot)}(\mathsf{pk})[[\tau]]$
  $c_l \leftarrow \mathsf{enc}(\mathsf{pk}, x_l)$
  $t \leftarrow \mathrm{Adv}_2(c_l)[[\tau]]$
until $t = \mathsf{true}$
return τ

$\mathbf{CFE\text{-}Exp}^{\mathbf{Ideal}}_{\mathcal{E},\mathbf{Adv},S}$

$(\mathsf{pk}, s) \leftarrow S_1()$
$(l, \tau) \leftarrow (0, 0)$
repeat
  $l \leftarrow l + 1$
  $x_l \leftarrow \mathrm{Adv}_1^{\mathcal{O}(\cdot,x_1,\ldots,x_{l-1})[[s]]}(\mathsf{pk})[[\tau]]$
  $(f_1, \ldots, f_q) \leftarrow$ all queries by Adv_1 so far
  $c_l \leftarrow S_3(f_0(x_l), \ldots, f_q(x_l))[[s]]$
  $t \leftarrow \mathrm{Adv}_2(c_l)[[\tau]]$
until $t = \mathsf{true}$
return τ

Figure 3: Experiments for the CFE security definition. The oracle $\mathcal{O}$ in the ideal experiment is defined as $\mathcal{O}(f, x_1, \ldots, x_{l-1})[[s]] \coloneqq S_2(f, f(x_1), \ldots, f(x_{l-1}))[[s]]$.

As mentioned before, this definition is close to a fully adaptive version of the one given in [BSW11]. One difference is that Adv_2 is not given oracle access to a key-generation oracle in our definition. This simplifies especially the ideal experiment and is not necessary here since Adv_2 can store its query in τ and Adv_1 can then query its oracle and continue the execution of Adv_2 in the following iteration. Furthermore, S_3 in [BSW11] has oracle access to Adv_2 and can therefore run Adv_2 on several inputs and discard undesired outputs. Our definition is stronger because we do not allow this. It was already noted in [AGVW13] that this oracle access might be problematic. Note that, in contrast to [BSW11, Definition 4], it is sufficient to return τ because Adv_2 can encode all relevant information in it and S_3 cannot tamper with it.

Unlike many other definitions (e.g., [O'N10, BF13, GVW12, GKP+13]), we do allow S_1 and S_2 to "fake" the public key and the secret keys, respectively. In contrast to what is claimed in [BF13], it turns out that this is not a problem (see Section 4.3 for more details). This shows that there are many degrees of freedom in defining security experiments for functional encryption and that the consequences of a particular choice are often unclear. On the other hand, the constructive approach we follow in this paper makes explicit what a protocol satisfying the definitions achieves, by specifying the resource that is constructed.

4.2 Equivalence of CFE-Security and Construction of Repository

The goal of this section is to prove that the protocol defined in Section 3.2 constructs the corresponding repository resource if and only if the underlying functional encryption scheme is CFE-secure. This implies that CFE-security is precisely the definition needed for our purpose. The following lemma shows that CFE-security is sufficient for the construction.

Lemma 4.2. *Let S_1, S_2, and S_3 be efficient probabilistic algorithms. Then there exists an efficient converter σ_B such that for all efficient distinguishers $\mathbf{D}$ for $\pi_A\pi_C[\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}]$ and $\sigma_B\mathsf{REP}_F$, there is an efficient probabilistic oracle algorithm Adv_1, an efficient probabilistic algorithm Adv_2, and an efficient distinguisher $\mathbf{D}'$ for the CFE experiment such that*

$$\Delta^{\mathbf{D}}\big(\pi_A\pi_C[\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}], \sigma_B\mathsf{REP}_F\big) = \Delta^{\mathbf{D}'}\big(\text{CFE-Exp}^{\text{Real}}_{\mathcal{E},\text{Adv}}, \text{CFE-Exp}^{\text{Ideal}}_{\mathcal{E},\text{Adv},S}\big).$$

Proof. We define σ_B as follows. See Figure 4 for a graphical overview of the simulator.

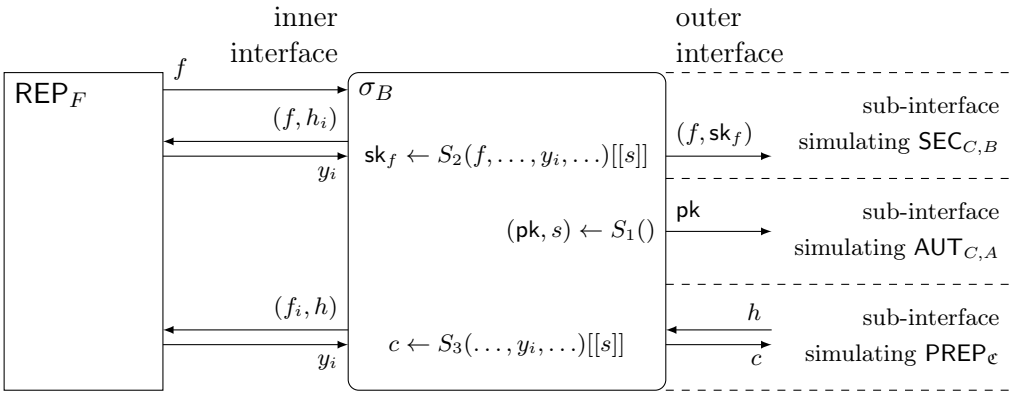


Figure 4: Overview of the simulator σ_B from the proof of Lemma 4.2 attached to REP_F. This figure corresponds to a refinement of the right part of Figure 1b. Lemma 4.2 implies that this resource is computationally indistinguishable from the resource one obtains by removing π_B from Figure 2 if the used functional encryption scheme is CFE-secure.

Initialization

$(l, q) \leftarrow (0, 0)$
$(\mathsf{pk}, s) \leftarrow S_1()$
output pk at outer sub-interface simulating $\mathsf{AUT}_{C,A}$

Inner Interface

Input: $f \in F$
$q \leftarrow q + 1$
$f_q \leftarrow f$
for $i = 1, \ldots, l$ **do**
output (f, h_i) at inner interface, let y_i be the returned value
$\mathsf{sk}_f \leftarrow S_2(f, y_1, \ldots, y_l)[[s]]$
output (f, sk_f) at outer sub-interface simulating $\mathsf{SEC}_{C,B}$

Outer Interface

Input: $h \in H$
if $\exists k \in \{1, \ldots, l\} : h_k = h$ **then**
output c_k at outer sub-interface simulating $\mathsf{PREP}_{\mathfrak{C}}$
else if output (f_0, h) at inner interface is not ignored **then** ▷ some data is stored for handle h
$l \leftarrow l + 1$
$h_l \leftarrow h$
for $i = 0, \ldots, q$ **do**
output (f_i, h) at inner interface, let y_i be the returned value
$c_l \leftarrow S_3(y_0, \ldots, y_q)[[s]]$
output c_l at outer sub-interface simulating $\mathsf{PREP}_{\mathfrak{C}}$

Now let $\mathbf{D}$ be an efficient distinguisher for the resources $\pi_A\pi_C[\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}]$ and $\sigma_B\mathsf{REP}_F$. We can assume without loss of generality that $\mathbf{D}$ inputs $h \in H$ at interface B only if this h was output at interface A before, because other h will be ignored by both resources. We further assume that each $h \in H$ is input at most once, since both resources return the same value for each input of the same handle h.

We now describe the algorithms Adv_1 and Adv_2. When Adv_1 is invoked with pk and $\tau = 0$, it starts a new simulation of the distinguisher $\mathbf{D}$, outputting pk at interface B from the authenticated channel. When $\mathbf{D}$ inputs $f \in F$ at interface C, Adv_1 invokes its oracle with query f and outputs f and the answer to $\mathbf{D}$ at interface B from the secure channel. When $\mathbf{D}$ inputs $x \in X$ at interface A, Adv_1 invokes `getHandle` and outputs the returned handle h at interface A. It further sets $M[h] \leftarrow x$ for a map M. When $\mathbf{D}$ inputs $h \in H$ at interface B, Adv_1 saves M and the state of $\mathbf{D}$ in τ and returns $M[h]$. When $\mathbf{D}$ returns a bit b, Adv_1 sets $\tau \leftarrow (\mathsf{return}, b)$ and returns a random $x \in X$. After Adv_1 terminated, Adv_2 is invoked on input a ciphertext c and τ. If $\tau = (\mathsf{return}, b)$, for some $b \in \{0, 1\}$, Adv_2 returns true. Otherwise, it saves c in τ and returns false. Afterwards, Adv_1 is invoked on input pk and $\tau \neq 0$. In this case, it reads c and the state of $\mathbf{D}$ from τ and continues the simulation by outputting c at interface B from the repository. Adv_1 then proceeds as above.

The distinguisher $\mathbf{D}'$ on input $\tau = (\mathsf{return}, b)$ outputs the bit b. Note that the distribution of the outputs of $\mathbf{D}'$ given outputs of the real experiment equals the distribution of the outputs of $\mathbf{D}$ connected to the resource $\pi_A \pi_C[\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}]$ and the output distribution of $\mathbf{D}'$ given outputs of the ideal experiment equals the distribution of the outputs of $\mathbf{D}$ connected to $\sigma_B \mathsf{REP}_F$. Hence, the corresponding distinguishing advantages are the same. □

The next lemma shows that CFE-security is not only sufficient but also necessary for constructions of the desired repository resource. Since this notion is even stronger than [BSW11, Definition 4], known impossibility results translate to our model.

Lemma 4.3. *Let σ_B be an efficient converter. Then there exist efficient probabilistic algorithms S_1, S_2, and S_3 such that for all efficient probabilistic oracle algorithms Adv_1, all efficient probabilistic algorithms Adv_2, and all efficient distinguishers $\mathbf{D}$ for the CFE experiment, there exists an efficient distinguisher $\mathbf{D}'$ for $\pi_A \pi_C[\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}]$ and $\sigma_B \mathsf{REP}_F$ such that*

$$\Delta^{\mathbf{D}}\big(\text{CFE-Exp}^{\mathrm{Real}}_{\mathcal{E},\mathrm{Adv}}, \text{CFE-Exp}^{\mathrm{Ideal}}_{\mathcal{E},\mathrm{Adv},S}\big) = \Delta^{\mathbf{D}'}\big(\pi_A \pi_C[\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}], \sigma_B \mathsf{REP}_F\big).$$

Proof. The algorithms S_1, S_2, and S_3 together simulate an execution of σ_B. S_1 starts the simulation, prepares an initially empty map M (i.e., $M[f][h] = \bot$ for all f and h), and sets $(l, q) \leftarrow (0, 0)$. It returns the first output at the outer interface of σ_B together with an encoding of the state of σ_B, l, q, and M in s. On input $f \in F$, $f(x_1), \ldots, f(x_l)$ and some s, S_2 extracts M, l, q, $h_1, \ldots, h_l$, and the state of σ_B from s, sets $q \leftarrow q + 1$, $f_q \leftarrow f$, and $M[f_q][h_i] \leftarrow f(x_i)$ for $i = 1, \ldots, l$. It then inputs f at the inner interface of σ_B. When σ_B outputs (f, sk_f) at its outer interface, S_2 stores the state of σ_B together with M, q, and f_q in s and returns sk_f. On input $(f_0(x), \ldots, f_q(x))$ and s, S_3 extracts the state of σ_B, M, l, q, and $f_1, \ldots, f_q$ from s, sets $l \leftarrow l + 1$, invokes $h_l \leftarrow$ `getHandle` (if `getHandle` requires interaction with interface A, S_3 emulates it using an arbitrary fixed strategy), sets $M[f_i][h_l] \leftarrow f_i(x)$ for $i = 0, \ldots, q$, and inputs h at the outer sub-interface of σ_B simulating the repository. When σ_B outputs c at its outer interface, S_3 saves the state of σ_B, M, l, and h_l in s and returns c. Outputs of the form (f, h) at the inner interface of σ_B are handled equally by S_1, S_2, and S_3 by inputting $M[f][h]$ at its inner interface if $M[f][h] \neq \bot$. Otherwise, that input is ignored. Note that such input is ignored if and only if f has not been input at the inner interface of σ_B or h has not been input at its outer interface before. This is consistent with REP_F if all handles returned at interface A are immediately input at interface B afterwards. The distinguisher defined below always does this.

Now let Adv_1 be an efficient probabilistic oracle algorithm, Adv_2 an efficient probabilistic algorithm, and let $\mathbf{D}$ be an efficient distinguisher for the CFE experiment. We define a

distinguisher $\mathbf{D}'$ for the resources $\pi_A\pi_C[\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}]$ and $\sigma_B\mathsf{REP}_F$ as follows. It first runs Adv_1 on input the initial output pk at interface B and $\tau = 0$. A query $f \in F$ from Adv_1 to its oracle is answered by inputting f at interface C, receiving (f, sk_f) at interface B, and returning sk_f as the answer. When Adv_1 returns x, $\mathbf{D}'$ inputs x at interface A and then the returned handle h at interface B to the repository (where $\mathbf{D}'$ uses the same strategy as S_3 for inputs at interface A for `getHandle`, such that handles are equally distributed). When c is output at interface B, the distinguisher $\mathbf{D}'$ invokes Adv_2 on input c and τ. If it returns $t = \mathsf{false}$, $\mathbf{D}'$ repeats this procedure by running Adv_1 on input pk and τ. Otherwise, $\mathbf{D}'$ invokes $\mathbf{D}$ on input τ. Finally, $\mathbf{D}'$ outputs the output of $\mathbf{D}$. Since the distribution of τ if $\mathbf{D}'$ is connected to $\pi_A\pi_C[\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}]$ is the same as in the real CFE experiment and the same as in the ideal one if $\mathbf{D}'$ is connected to $\sigma_B\mathsf{REP}_F$, the corresponding distinguishing advantages are equal. □

Combining lemmata 3.3, 4.2, and 4.3, we get the following theorem:

Theorem 4.4. *For the protocol π defined above, we have*

$$[\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}] \overset{\pi}{\Longrightarrow} \mathsf{REP}_F \qquad \Longleftrightarrow \qquad \mathcal{E} \textit{ is CFE-secure.}$$

4.3 Alleged Insufficiency of BSW's Security Definition

The results from the previous section seem to contradict an example given in [BF13], which was meant to show that [BSW11, Definition 4] is not adequate and which can easily be extended to our definition. We first recall the example for a fixed set of functions from the full version of [BF13] and then explain why it is not a problem in our context. Assume $\mathcal{E} = (\mathsf{setup}, \mathsf{keygen}, \mathsf{enc}, \mathsf{dec})$ is a functional encryption scheme for a set F of functions with domain X with $\mathrm{id}_X \in F$ and $P \subseteq F$ where P is a family of trapdoor one-way permutations on X. We consider a modified scheme $\mathcal{E}' = (\mathsf{setup}', \mathsf{keygen}', \mathsf{enc}, \mathsf{dec}')$ as follows. The algorithm setup' first runs setup and samples a permutation $p^* \in P$ according to the key-generation algorithm of the trapdoor one-way permutation. It then includes the description of p^* in the public key $\mathsf{pk}' := (\mathsf{pk}, p^*)$ and the master secret key $\mathsf{mk}' := (\mathsf{mk}, p^*)$ (and discards the trapdoor). The algorithm keygen' on input (mk', f) does the same as keygen if $f \neq p^*$, and returns $(p^*, \mathsf{sk}_{\mathrm{id}})$ with $\mathsf{sk}_{\mathrm{id}} \leftarrow \mathsf{keygen}(\mathsf{mk}, \mathrm{id}_X)$ if $f = p^*$. The algorithm enc is not modified and dec' on input $((p^*, \mathsf{sk}_{\mathrm{id}}), c)$ returns $p^*(\mathsf{dec}(\mathsf{sk}_{\mathrm{id}}, c))$ and $\mathsf{dec}(\mathsf{sk}_f, c)$ on input (sk_f, c). As in [BF13], it can be shown that $\mathcal{E}'$ is CFE-secure if $\mathcal{E}$ is. Intuitively, the simulator, which generates the public key, can store the trapdoor and hence compute x from $p^*(x)$, enabling it to simulate.

According to [BF13], this scheme should not be considered secure because one can learn x instead of only $p^*(x)$ given a key for p^*. They conclude that the simulator should therefore not be allowed to generate the public key. We claim that this is actually not a problem: An adversary cannot choose for which $p \in P$ he wants to learn x instead of $p(x)$, but a p^* is chosen at random by the key-generation algorithm of the trapdoor one-way permutation. By simply invoking this algorithm themselves, all users can obtain a trapdoor for such a random permutation p^* and hence compute x from $p^*(x)$ even if the original scheme is used. The only difference is that in the modified scheme, this particular permutation is contained in the public key and the master secret key. Using this permutation in another protocol built on top of the modified scheme would be problematic if the designer of the composed protocol assumes that one cannot compute x if given a key for this p^* (which is the case in the original scheme if the description of a random p^* is included in the public key, but not in the modified scheme).

However, if schemes are composed according to the constructive cryptography framework, such confusion cannot arise since the protocol converters use the keys only internally and do not publish them to their outer interfaces. Hence, the constructed resource does not provide a distinguished function corresponding to the permutation in the public key to any party. This means that when the constructed resource is used in another protocol, that protocol cannot explicitly use this particular p^*. Because $p^* \in P$ is chosen randomly and P needs to be large, the probability that it is still used in another context is negligible. Therefore the modified scheme is as secure as the original one in all applications if the protocols are composed properly.

5 Special Cases and Impossibility Results

5.1 Public-Key Encryption and its Impossibility

As described in [BSW11], many types of encryption can be seen as special cases of functional encryption. We restate how standard public-key encryption is captured as a special case and explain why this immediately leads to strong impossibility results.

Consider public-key encryption with plaintext space M. We can set $X \coloneqq M$ and $F_{\mathrm{PKE}} \coloneqq \{f_0, \mathrm{id}_X\}$ with $f_0 \colon X \to \mathbb{N}, x \mapsto |x|$ revealing the bit length of x. This provides the same functionality as public-key encryption [BSW11]: The holder of the secret key (corresponding to the key for id_X) can decrypt a ciphertext to the encrypted message while without the secret key, one can only learn the length of the message.

Depending on how the encryption scheme is intended to be used, different security properties are required. Typically, one assumes that there is a legitimate receiver Bob knowing the secret key from the beginning and an eavesdropper Eve who never learns the secret key. The repository resource $\mathsf{REP}_{F_{\mathrm{PKE}}}$, however, allows to adaptively grant Bob the right to learn the input data. In case of a dishonest Bob, this enables the distinguisher to adaptively retrieve the secret key after receiving ciphertexts. Hence, we are in a situation of adaptive adversaries [CFGN96]. Therefore, the result by Nielsen [Nie02] stating that the length of secret keys in any adaptively secure scheme must be at least the total number of bits to be encrypted, on which the impossibility results in [BSW11, BO13] are based, can be applied directly here. Since there is no restriction on the number of messages Alice can input in the repository, the distinguisher can input messages whose total length exceed an upper bound on the length of secret keys before granting access rights to Bob. We therefore get the following theorem as a direct consequence of Theorem 4.4 and the result from [Nie02]. This shows another advantage of our constructive approach, namely that defining security of a protocol by what it achieves simplifies reusing results without reproving them for new definitions as in [BSW11] and [BO13].

Theorem 5.1. *There is no CFE-secure functional encryption scheme for F_{PKE}.*

As we have seen, while public-key encryption can be considered to be a special case of functional encryption, typical security definitions for public-key encryption do not correspond to the given security definition for functional encryption. This is not a weakness of our security definition but comes from the fact that public-key encryption is usually used in a specific setting and thus a less restrictive definition is sufficient. See [CMT13] for a treatment of public-key encryption in the constructive cryptography framework. Similarly, it is still meaningful to consider security definitions for other special cases of functional encryption such as identity-based and attribute-based encryption that take into account how these schemes are intended to be used.

5.2 Circumventing Impossibility Results

As seen in the previous section, realizing CFE-secure functional encryption schemes, even for very simple sets of functions, is impossible without further assumptions. In general, there are two ways to circumvent such impossibility results. One can either start with a stronger assumed resource or construct a weaker resource. The authors in [BSW11] use a random oracle to realize secure functional encryption schemes for a large class of functions. This falls in the first category and can be understood in our model as adding a random oracle to the assumed resource. In Section 6, we recast the scheme from [BSW11] in our framework and show that it can be used to construct REP_F from $[\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}, \mathsf{RO}]$. Bellare and O'Neill [BO13] generalize the result from [BSW11] and obtain secure functional encryption schemes without random oracles by allowing the keys to be longer than all encrypted messages together. This can also be interpreted as restricting the amount of data the repository can store. Hence, this is an example of weakening the constructed resource.

Many papers consider different security definitions [O'N10,BO13,GVW12,AGVW13,GKP+13, GGH+13] that are not subject to impossibility results. However, changing a security definition can lead to an inadequate notion of security and it might not be clear how the resulting schemes can be used. A result from [CIJ+13] shows how to transform a functional encryption scheme for the set of all n-input Boolean circuits satisfying a weak security definition to a scheme that satisfies a stronger security definition in the random oracle model. This provides a bridge between considering weaker definitions and using stronger assumed resources (i.e., including random oracles) to achieve strong security notions.

In the Section 7, we follow the approach of constructing weaker resources. In particular, we introduce restricted repositories and show that schemes satisfying a definition from [GVW12] can be used to construct such repositories.

6 Construction in the Random Oracle Model

We recast the scheme presented in [BSW11] as "the modified 'brute-force' construction" in our framework. It allows us to construct REP_F for all F that contain only a polynomial (in the security parameter) number of functions that all have domain X and codomain $\{0,1\}^k$ for some $k \in \mathbb{N}$. The assumed resource in this construction contains a random oracle with codomain $\{0,1\}^k$. We now define such a random oracle as a resource.

Definition 6.1. The resource RO has interfaces A, B, and C. On input $x \in \{0,1\}^*$ at interface $I \in \{A,B,C\}$, if x has not been input before (at any interface), RO chooses $y \in \{0,1\}^k$ uniformly at random, outputs y at interface I, and stores (x,y) internally; if x has been input before, RO outputs the value y at interface I that has been stored together with x.

Remark. The simulator in the proof of the construction will answer queries to the random oracle made by the distinguisher. Since the simulator can answer these queries arbitrarily as long as they are consistent with previous answers and appear to be uniform, the simulator has additional power, which allows us to overcome the impossibility result from Section 5.1. This setting is often referred to as *programmable random oracle model* because the simulator can in some sense "reprogram" the random oracle.

Now let X be some nonempty set and $F = (f_0, \ldots, f_s)$ with $f_i \colon X \to \{0,1\}^k$. Further let (K,E,D) be a semantically secure public-key encryption scheme. Let $\mathfrak{C}$ be the set of $s+1$-tuples

where the first component is a k-bit string and the other components consist of an element from the range of E and a k-bit string. We define the protocol $\pi^{\mathrm{RO}} = (\pi_A^{\mathrm{RO}}, \pi_B^{\mathrm{RO}}, \pi_C^{\mathrm{RO}})$ to construct REP_F from $[\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}, \mathsf{RO}]$ as follows: At the beginning, π_C^{RO} invokes $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow K()$ for $i \in \{1, \ldots, s\}$, stores the sk_i and sends $(\mathsf{pk}_1, \ldots, \mathsf{pk}_s)$ to Alice over the authenticated channel. The public keys are internally stored by π_A^{RO}. On input $x \in X$ at its outer interface, π_A^{RO} chooses $r_1, \ldots, r_s$ uniformly at random from $\{0,1\}^\lambda$ where λ is the security parameter. It then outputs r_i at its inner interface to the random oracle and receives the answer r'_i for each $i \in \{1, \ldots, s\}$. Finally, it outputs $c \leftarrow (f_0(x), (E(\mathsf{pk}_1, r_1), r'_1 \oplus f_1(x)), \ldots, (E(\mathsf{pk}_s, r_s), r'_s \oplus f_s(x)))$ at its inner interface to the repository and outputs the returned handle h at its outer interface. On input $f_i \in F$ at its outer interface, π_C^{RO} sends (f_i, sk_i) to B over the secure channel. The converter π_B^{RO} then stores the secret key and outputs f_i at its outer interface. On input $(f_i, h) \in F \times H$ at its outer interface, π_B^{RO} outputs h at its inner interface to the repository if it has stored the secret key sk_i or if $i = 0$. If it receives $c = (c_0, \ldots, c_s)$ from the repository, it outputs c_0 at its outer interface if $i = 0$, and if $i \neq 0$, it decrypts the first component of c_i with $D(\mathsf{sk}_i, \cdot)$ and outputs the result at its inner interface to RO. When RO answers with r, π_B^{RO} outputs the bitwise XOR of r and the second component of c_i at its outer interface. All other inputs are ignored.

Theorem 6.2. *We have*

$$[\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}, \mathsf{RO}] \overset{\pi^{\mathrm{RO}}}{\Longrightarrow} \mathsf{REP}_F.$$

Proof sketch. Correctness is straightforward to verify. To prove the security condition, we consider the simulator σ_B that initially generates $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow K()$ for $i \in \{1, \ldots, s\}$ and outputs $(\mathsf{pk}_1, \ldots, \mathsf{pk}_s)$ at its outer sub-interface simulating $\mathsf{AUT}_{C,A}$. To answer queries to the simulated random oracle, the simulator maintains a list O of all previous inputs and outputs. Queries in O are answered consistently. For a fresh input, the simulator chooses a value uniformly at random from $\{0,1\}^k$, outputs this value at its outer sub-interface simulating RO, and updates O. To simulate $\mathsf{PREP}_{\mathfrak{C}}$, as the simulator in the proof of Lemma 4.2, σ_B keeps track of all queried handles and the generated ciphertexts, answers repeated queries consistently, and ignores queries for invalid handles. A fresh ciphertext for handle h is generated as $(y_0, (E(\mathsf{pk}_1, r_1), r'_1), \ldots, (E(\mathsf{pk}_s, r_s), r'_s))$, where y_0 is obtained from REP_F via the query (f_0, h) and $r_1, \ldots, r_s \in \{0,1\}^\lambda$, $r'_1, \ldots, r'_s \in \{0,1\}^k$ are chosen uniformly at random. Moreover, the simulator obtains y_i from REP_F via the query (f_i, h) for all f_i for which access has been granted, and adds $(r_i, r'_i \oplus y_i)$ to O.

On input $f_i \in F$ at its inner interface, σ_B outputs (f_i, sk_i) at its outer sub-interface simulating $\mathsf{SEC}_{C,B}$. Furthermore, it obtains $y_{j,i}$ from REP_F via the query (f_i, h_j) for all h_j for which ciphertexts have been simulated. Let $(y_{j,0}, (E(\mathsf{pk}_1, r_{j,1}), r'_{j,1}), \ldots, (E(\mathsf{pk}_s, r_{j,s}), r'_{j,s}))$ be these ciphertexts. Finally, σ_B adds $(r_{j,i}, r'_{j,i} \oplus y_{j,i})$ to O. Whenever an entry (x, y) is supposed to be added to O and this list already contains an entry for x, the simulator aborts.

Note that if the simulator does not abort, if all r_i chosen by σ_B are unique and different from previous oracle queries by the distinguisher, and if the distinguisher does not query the random oracle on a value r such that $E(\mathsf{pk}_i, r)$ is contained in some ciphertext generated by σ_B for some i, the simulation is perfect. Since (K, E, D) is semantically secure, it can be shown that the probability of these events is negligible. Thus, $\pi_A^{\mathrm{RO}} \pi_C^{\mathrm{RO}} [\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}, \mathsf{RO}]$ and $\sigma_B \mathsf{REP}_F$ are computationally indistinguishable. □

7 Weaker Security Definitions and Restricted Variants of Repository Resources

7.1 Definitions

We now define restricted variants of repository resources, which may still be sufficient for certain applications and can then be used as sketched in Appendix A.

Definition 7.1. Let $L, Q \in \mathbb{N} \cup \{\infty\}$, X be a nonempty set, and let F be a set of functions with domain X and $f_0 \in F$. We define the resource $\mathsf{REP}^{\mathrm{AD}}_{F,L,Q}$ to be identical to REP_F as in Definition 3.1 but only allow up to L inputs at interface A and Q inputs at interface C and ignore further inputs (in case L or Q equals ∞, no restriction is placed on the number of inputs, i.e., $\mathsf{REP}^{\mathrm{AD}}_{F,\infty,\infty} \equiv \mathsf{REP}_F$). We further define a nonadaptive variant $\mathsf{REP}^{\mathrm{NA}}_{F,L,Q}$ that ignores all inputs at interface C after the first input at interface A.

Since the resources only accept a given number of inputs at interfaces A and C, we have to adjust the protocols to behave the same way. We therefore consider a functional encryption scheme $\mathcal{E} = (\mathsf{setup}, \mathsf{keygen}, \mathsf{enc}, \mathsf{dec})$ for a set F of functions with domain X where $\mathfrak{C}$ is the range of enc. We then define the protocol $\pi^{\mathrm{AD},L,Q} := (\pi^L_A, \pi_B, \pi^{\mathrm{AD},Q}_C)$ as π in Section 3.2, but π^L_A and $\pi^{\mathrm{AD},Q}_C$ additionally keep track of the number of inputs at their outer interface and ignore all of them after the first L and Q inputs, respectively.

The following lemma states that this protocol yields a restricted repository with access control if all parties are honest. It follows directly from the correctness of the functional encryption scheme.

Lemma 7.2. *For the protocol $\pi^{\mathrm{AD},L,Q} = (\pi^L_A, \pi_B, \pi^{\mathrm{AD},Q}_C)$ defined above, we have*

$$\pi^L_A \pi_B \pi^{\mathrm{AD},Q}_C[\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}] \;\approx\; \mathsf{REP}^{\mathrm{AD}}_{F,L,Q}.$$

To construct the nonadaptive variant, the protocol at interface C has to ignore all inputs after the first input at interface A. Hence, it needs to know whether there has already been any input at interface A, i.e., whether the repository is still empty. We thus introduce for a nonempty set X the resource $\mathsf{PREP}^{\emptyset}_X$ that is identical to PREP_X as in Definition 3.2 but additionally accepts the input $\mathsf{isEmpty}$ at interface C which is answered with true if the repository is empty, and false otherwise. We then define $\pi^{\mathrm{NA},L,Q} := (\pi^L_A, \pi_B, \pi^{\mathrm{NA},Q}_C)$ where $\pi^{\mathrm{NA},Q}_C$ on each input at its outer interface outputs $\mathsf{isEmpty}$ at its inner interface to the repository and ignores the input (and all subsequent inputs) if the repository answers false, and otherwise does the same as $\pi^{\mathrm{AD},Q}_C$.

As above, correctness of the functional encryption scheme implies the following lemma.

Lemma 7.3. *For the protocol $\pi^{\mathrm{NA},L,Q} = (\pi^L_A, \pi_B, \pi^{\mathrm{NA},Q}_C)$ defined above, we have*

$$\pi^L_A \pi_B \pi^{\mathrm{NA},Q}_C[\mathsf{PREP}^{\emptyset}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}] \;\approx\; \mathsf{REP}^{\mathrm{NA}}_{F,L,Q}.$$

Having defined these resources and protocols, we can derive the following security definitions.

Definition 7.4. Let $\mathcal{E} = (\mathsf{setup}, \mathsf{keygen}, \mathsf{enc}, \mathsf{dec})$ be a functional encryption scheme for a set F of functions with domain X where $\mathfrak{C}$ is the range of enc and let $L, Q \in \mathbb{N} \cup \{\infty\}$. We say $\mathcal{E}$ is (L, Q)-*AD-CFE-secure* if

$$[\mathsf{PREP}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}] \;\overset{\pi^{\mathrm{AD},L,Q}}{\Longrightarrow}\; \mathsf{REP}^{\mathrm{AD}}_{F,L,Q},$$

$\mathbf{NA\text{-}SIM\text{-}Exp}^{\mathbf{Real}}_{\mathcal{E},\mathbf{Adv}}$	$\mathbf{NA\text{-}SIM\text{-}Exp}^{\mathbf{Ideal}}_{\mathcal{E},\mathbf{Adv},\boldsymbol{S}}$
$(\mathsf{pk},\mathsf{mk}) \leftarrow \mathsf{setup}()$	$(\mathsf{pk},\mathsf{mk}) \leftarrow \mathsf{setup}()$
$(x,\tau) \leftarrow \mathrm{Adv}_1^{\mathsf{keygen}(\mathsf{mk},\cdot)}(\mathsf{pk})$	$(x,\tau) \leftarrow \mathrm{Adv}_1^{\mathsf{keygen}(\mathsf{mk},\cdot)}(\mathsf{pk})$
	$(f_1,\ldots,f_q) \leftarrow$ oracle queries by Adv_1
	$(\mathsf{sk}_1,\ldots\mathsf{sk}_q) \leftarrow$ replies from keygen oracle
	$(y_0,\ldots,y_q) \leftarrow (f_0(x),\ldots,f_q(x))$
$c \leftarrow \mathsf{enc}(\mathsf{pk},x)$	$c \leftarrow S(\mathsf{pk},f_1,\ldots,f_q,\mathsf{sk}_1,\ldots\mathsf{sk}_q,y_0,\ldots,y_q)$
$\alpha \leftarrow \mathrm{Adv}_2(\mathsf{pk},c,\tau)$	$\alpha \leftarrow \mathrm{Adv}_2(\mathsf{pk},c,\tau)$
return (α,x)	**return** (α,x)

Figure 5: Experiments for the Q-NA-SIM security definition. Note that $q \leq Q$ for Adv_1 that make at most Q queries.

and $\mathcal{E}$ is (L,Q)-*NA-CFE-secure* if

$$[\mathsf{PREP}^{\emptyset}_{\mathfrak{C}}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}] \overset{\pi^{\mathrm{NA},L,Q}}{\Longmapsto} \mathsf{REP}^{\mathrm{NA}}_{F,L,Q}.$$

Note that CFE-security corresponds to (∞,∞)-AD-CFE-security by Theorem 4.4. We now recall a simulation-based single-message security definition from [GVW12], which we will compare to our new notions in the next section.

Definition 7.5. Let $\mathcal{E} = (\mathsf{setup},\mathsf{keygen},\mathsf{enc},\mathsf{dec})$ be a functional encryption scheme for a set F of functions with common domain X. We introduce the two experiments in Figure 5 for an efficient probabilistic oracle algorithm Adv_1 and efficient probabilistic algorithms Adv_2 and S. The advantage of a distinguisher $\mathbf{D}$ in distinguishing the outputs of these experiments is denoted by $\Delta^{\mathbf{D}}\left(\text{NA-SIM-Exp}^{\mathrm{Real}}_{\mathcal{E},\mathrm{Adv}}, \text{NA-SIM-Exp}^{\mathrm{Ideal}}_{\mathcal{E},\mathrm{Adv},S}\right)$.

For $Q \in \mathbb{N}$, the scheme $\mathcal{E}$ is Q-*NA-SIM-secure* if there exists an efficient probabilistic algorithm S such that $\Delta^{\mathbf{D}}\left(\text{NA-SIM-Exp}^{\mathrm{Real}}_{\mathcal{E},\mathrm{Adv}}, \text{NA-SIM-Exp}^{\mathrm{Ideal}}_{\mathcal{E},\mathrm{Adv},S}\right)$ is negligible for all efficient probabilistic oracle algorithms Adv_1 that make at most Q queries, all efficient probabilistic algorithms Adv_2, and for all efficient distinguishers $\mathbf{D}$.

7.2 Sufficiency of NA-SIM-Security

In this section, we show that Q-NA-SIM-security is sufficient to construct a nonadaptive repository with potentially dishonest B. This shows that the scheme constructed in [GVW12], which satisfies this definition, can be used in a composable framework. In particular, this shows how to construct a nonadaptive repository for the set of all functions that are computed by polynomial-size circuits, when the number of granted access rights is bounded.

Note that in contrast to Definition 4.1, all keys the adversary sees in the ideal experiment are generated by the algorithms of the functional encryption scheme and not by a simulator. While this is an artificial restriction for constructing single-input repositories, it interestingly allows us to prove that this definition is sufficient to construct a repository for many inputs. A similar result was already shown in [GVW12] but our result is stronger because we allow subsequent inputs to depend on previous ciphertexts whereas the many-message definition in [GVW12] restricts the adversary to input all messages at once before seeing a ciphertext.

Lemma 7.6. *Let $L,Q \in \mathbb{N}$ and let S be an efficient probabilistic algorithm. Then there exists an efficient converter σ_B such that for all efficient distinguishers $\mathbf{D}$ for the resources*

$\pi_A^L \pi_C^{\mathrm{NA},Q}[\mathsf{PREP}_{\mathfrak{C}}^{\emptyset}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}]$ *and* $\sigma_B \mathsf{REP}_{F,L,Q}^{\mathrm{NA}}$*, there is an efficient probabilistic oracle algorithm* Adv_1 *that makes at most* Q *queries, an efficient probabilistic algorithm* Adv_2*, and an efficient distinguisher* $\mathbf{D}'$ *for the NA-SIM experiment such that*

$$\Delta^{\mathbf{D}}\left(\pi_A^L \pi_C^{\mathrm{NA},Q}[\mathsf{PREP}_{\mathfrak{C}}^{\emptyset}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}], \sigma_B \mathsf{REP}_{F,L,Q}^{\mathrm{NA}}\right) = L \cdot \Delta^{\mathbf{D}'}\left(\text{NA-SIM-Exp}_{\mathcal{E},\mathrm{Adv}}^{\mathrm{Real}}, \text{NA-SIM-Exp}_{\mathcal{E},\mathrm{Adv},S}^{\mathrm{Ideal}}\right).$$

Proof. We define σ_B as follows:

Initialization

$(l, q) \leftarrow (0, 0)$
$(\mathsf{pk}, \mathsf{mk}) \leftarrow \mathsf{setup}()$
output pk at outer sub-interface simulating $\mathsf{AUT}_{C,A}$

Inner Interface

Input: $f \in F$
$q \leftarrow q + 1$
$f_q \leftarrow f$
$\mathsf{sk}_q \leftarrow \mathsf{keygen}(\mathsf{mk}, f)$
output (f, sk_q) at outer sub-interface simulating $\mathsf{SEC}_{C,B}$

Outer Interface

Input: $h \in H$
if $\exists k \in \{1, \ldots, l\} : h_k = h$ **then**
output c_k at outer sub-interface simulating $\mathsf{PREP}_{\mathfrak{C}}^{\emptyset}$
else if output (f_0, h) at inner interface is not ignored **then** ▷ some data stored for handle h
$l \leftarrow l + 1$
$h_l \leftarrow h$
for $i = 0, \ldots, q$ **do**
output (f_i, h) at inner interface, let y_i be the returned value
$c_l \leftarrow S(\mathsf{pk}, f_1, \ldots, f_q, \mathsf{sk}_1, \ldots \mathsf{sk}_q, y_0, \ldots, y_q)$
output c_l at outer sub-interface simulating $\mathsf{PREP}_{\mathfrak{C}}^{\emptyset}$

Now let $\mathbf{D}$ be an efficient distinguisher for the resources $\pi_A^L \pi_C^{\mathrm{NA},Q}[\mathsf{PREP}_{\mathfrak{C}}^{\emptyset}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}]$ and $\sigma_B \mathsf{REP}_{F,L,Q}^{\mathrm{NA}}$. We can assume that $\mathbf{D}$ does not make any inputs that are ignored by both resources. Hence, we can assume that after getting a public key from interface B, $\mathbf{D}$ makes up to Q inputs of the form $f \in F$ at interface C. Afterwards, it makes inputs of the form $x \in X$ at interface A and $h \in H$ at interface B for h that were output at interface A before. As in the proof of Lemma 4.2, we can also assume without loss of generality that each h is input at most once, because both resources return the same value for each input of the same h.

We let Adv_1, Adv_2, and $\mathbf{D}'$ emulate $\mathbf{D}$. At the beginning, Adv_1 sets $l \leftarrow 0$, draws a number $\hat{l} \in \{1, \ldots, L\}$ uniformly at random and outputs pk at interface B from the authenticated channel for $\mathbf{D}$. When $\mathbf{D}$ inputs $f \in F$ at interface C, Adv_1 makes the oracle-query f and outputs f and the answer sk at interface B from the secure channel. When $\mathbf{D}$ inputs x at interface A, Adv_1 invokes `getHandle` and outputs the returned handle h at interface A. It further sets $M[h] \leftarrow x$ for a map M. When $\mathbf{D}$ inputs $h \in H$ at interface B, Adv_1 increments l by one. If $l < \hat{l}$, Adv_1 outputs $\mathsf{enc}(\mathsf{pk}, M[h])$ at interface B from the repository. If $l \geq \hat{l}$, Adv_1 saves M, the list $f_1, \ldots, f_q$ of queried functions, the answers $\mathsf{sk}_1, \ldots, \mathsf{sk}_q$, and the state of $\mathbf{D}$

in τ and returns $(M[h], \tau)$. On input (pk, c, τ), Adv_2 reads M, $f_1, \ldots, f_q$, $\mathsf{sk}_1, \ldots, \mathsf{sk}_q$, and the state of $\mathbf{D}$ from τ and continues the simulation of $\mathbf{D}$ by outputting c at interface B from the repository. When $\mathbf{D}$ inputs x at interface A, Adv_2 invokes `getHandle`, outputs the returned handle h at interface A, and sets $M[h] \leftarrow x$. When $\mathbf{D}$ inputs $h \in H$ at interface B, Adv_2 computes $y_0 \leftarrow f_0(M[h]), \ldots, y_q \leftarrow f_q(M[h])$ and $c' \leftarrow S(\mathsf{pk}, f_1, \ldots, f_q, \mathsf{sk}_1, \ldots \mathsf{sk}_q, y_0, \ldots, y_q)$ and outputs c' at interface B from the repository. When $\mathbf{D}$ outputs a bit, Adv_2 returns this bit. The distinguisher $\mathbf{D}'$ on input (α, x) simply outputs α.

We prove the bound on the distinguishing advantage by a hybrid argument. To this end, consider for $i = 0, \ldots, L$ the system $\mathbf{H}_i$ that corresponds to $\pi_A^L \pi_C^{\mathrm{NA},Q}[\mathsf{PREP}_{\mathfrak{C}}^{\emptyset}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}]$ for the first i inputs of the form $h \in H$ at interface B, and that subsequently on input $h \in H$ at interface B behaves as follows: If h has been output at interface A but not input at interface B before, the resource outputs $S(\mathsf{pk}, f_1, \ldots, f_q, \mathsf{sk}_1, \ldots \mathsf{sk}_q, y_0, \ldots, y_q)$ at interface B, where q is the number of inputs at interface C, f_j corresponds to the jth input at interface C, sk_j to the value the resource output in return at interface B together with f_j, and $y_j = f_j(x)$ for the $x \in X$ that was input at interface A before the resource returned h. Inputs $h \in H$ at interface B for h that have not been output before at interface A are ignored, and inputting the same h more than once always yields the same output as after its initial input. Note that we have $\mathbf{H}_L \equiv \pi_A^L \pi_C^{\mathrm{NA},Q}[\mathsf{PREP}_{\mathfrak{C}}^{\emptyset}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}]$ and $\mathbf{H}_0 \equiv \sigma_B \mathsf{REP}_{F,L,Q}^{\mathrm{NA}}$. Further note that

$$\begin{aligned}
\mathrm{P}\big(\mathbf{D}' \text{ NA-SIM-Exp}_{\mathcal{E},\mathrm{Adv}}^{\mathrm{Real}} = 1\big) &= \sum_{i=1}^{L} \mathrm{P}\big(\hat{l} = i\big) \cdot \mathrm{P}\big(\mathbf{D}' \text{ NA-SIM-Exp}_{\mathcal{E},\mathrm{Adv}}^{\mathrm{Real}} = 1 \mid \hat{l} = i\big) \\
&= \frac{1}{L} \sum_{i=1}^{L} \mathrm{P}(\mathbf{DH}_i = 1)
\end{aligned}$$

and similarly

$$\mathrm{P}\big(\mathbf{D}' \text{ NA-SIM-Exp}_{\mathcal{E},\mathrm{Adv},S}^{\mathrm{Ideal}} = 1\big) = \frac{1}{L} \sum_{i=1}^{L} \mathrm{P}(\mathbf{DH}_{i-1} = 1).$$

We therefore have

$$\begin{aligned}
&\Delta^{\mathbf{D}}\left(\pi_A^L \pi_C^{\mathrm{NA},Q}[\mathsf{PREP}_{\mathfrak{C}}^{\emptyset}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}], \sigma_B \mathsf{REP}_{F,L,Q}^{\mathrm{NA}}\right) \\
&= \left|\mathrm{P}(\mathbf{DH}_L = 1) - \mathrm{P}(\mathbf{DH}_0 = 1)\right| \\
&= \left|\sum_{i=1}^{L} \mathrm{P}(\mathbf{DH}_i = 1) - \sum_{i=1}^{L} \mathrm{P}(\mathbf{DH}_{i-1} = 1)\right| \\
&= \left|L \cdot \mathrm{P}\big(\mathbf{D}' \text{ NA-SIM-Exp}_{\mathcal{E},\mathrm{Adv}}^{\mathrm{Real}} = 1\big) - L \cdot \mathrm{P}\big(\mathbf{D}' \text{ NA-SIM-Exp}_{\mathcal{E},\mathrm{Adv},S}^{\mathrm{Ideal}} = 1\big)\right| \\
&= L \cdot \Delta^{\mathbf{D}'}\left(\text{NA-SIM-Exp}_{\mathcal{E},\mathrm{Adv}}^{\mathrm{Real}}, \text{NA-SIM-Exp}_{\mathcal{E},\mathrm{Adv},S}^{\mathrm{Ideal}}\right). \qquad \square
\end{aligned}$$

Note that since an efficient distinguisher can only make a polynomial number of inputs at interface A, lemmata 7.3 and 7.6 imply the following theorem.

Theorem 7.7. *Let $Q \in \mathbb{N}$, $\mathcal{E}$ be a Q-NA-SIM-secure functional encryption scheme, and let $\pi^{\mathrm{NA},\infty,Q}$ be the protocol defined above for $\mathcal{E}$. We then have*

$$[\mathsf{PREP}_{\mathfrak{C}}^{\emptyset}, \mathsf{AUT}_{C,A}, \mathsf{SEC}_{C,B}] \quad \overset{\pi^{\mathrm{NA},\infty,Q}}{\Longmapsto} \quad \mathsf{REP}_{F,\infty,Q}^{\mathrm{NA}}.$$

Stated equivalently, Q-NA-SIM-security implies (∞, Q)-NA-CFE-security.

8 Capturing More General Notions of Functional Encryption

Recent papers generalize the notion of functional encryption to support functions of several variables [GGJS13, GKL+13, ABF+13] or randomized functions [GJKS15, GGJS13, ABF+13]. While a detailed treatment of these extensions is beyond the scope of this paper, we sketch how to capture them and further extensions in our model. Using our approach, one only needs to specify the involved resources; the corresponding security definitions are then implied by the constructive cryptography framework and one can extract equivalent traditional security definitions as we have done to obtain CFE-security. In contrast to that, adjusting a traditional security definition appropriately to support additional features is a challenging task and it is often unclear which guarantees a specific definition provides. Whether existing definitions are sufficient to achieve the constructions we propose in this section is an open question, but we conjecture that this is not the case.

8.1 Dishonest Senders

So far, we have assumed that Alice is always honest, that is in the real world, dishonest users do not encrypt data that is decrypted by someone else. Dropping this assumption is a very natural extension, which is indeed required for many applications. This scenario can be captured by simply considering dishonest users at interface A. According to a general principle of the constructive cryptography framework, security then means that interactions at interface A can also be simulated. Otherwise, all definitions in our paper remain unchanged. See [MR11] for more details. While in the context of randomized functions [GJKS15, GGJS13], security definitions have been considered that prevent the sender from tampering with the randomness, we are not aware of any results that provide guarantees against dishonest senders beyond preventing them from manipulating randomness.

8.2 Randomized Functions

It is straightforward to extend the repositories defined in Definition 7.1 to support randomized functions as follows: For a set F of functions with domain $R \times X$, the repository on input (f, h) at interface B chooses $r \in R$ uniformly at random and outputs $f(r, M[h])$ at interface B if the right to access f has been granted to B and $M[h] \neq \perp$. One can consider two cases here: Either the randomness r is chosen freshly for each input at interface B or the resource remembers all inputs and only samples r for fresh inputs (and uses the same randomness if the same pair (f, h) is input again). This can, of course, be combined with dishonest senders as explained above.

8.3 Functions of Several Variables

To compute on a database where each entry is encrypted separately, one needs to support functions f with domain X^n. Our constructed resources can be generalized to capture that by answering inputs $(f, h_1, \ldots, h_n) \in F \times H^n$ at interface B with $f(M[h_1], \ldots, M[h_n])$ if the right to access f has been granted. The setting considered in [GGJS13] is even more general: The functions there have domain $X_1 \times \ldots \times X_n$ and the key used to encrypt $x_i \in X_i$ can be different from the one used to encrypt $x_j \in X_j$ for $i \neq j$. To capture this setting, we extend our constructed resource to have interfaces $A_1, \ldots, A_n$ instead of interface A where elements in X_i can be input at interface A_i. Moreover, [GGJS13, GKL+13] allow some of the encryption keys to be secret while others are public. This can be covered easily in our model by adjusting

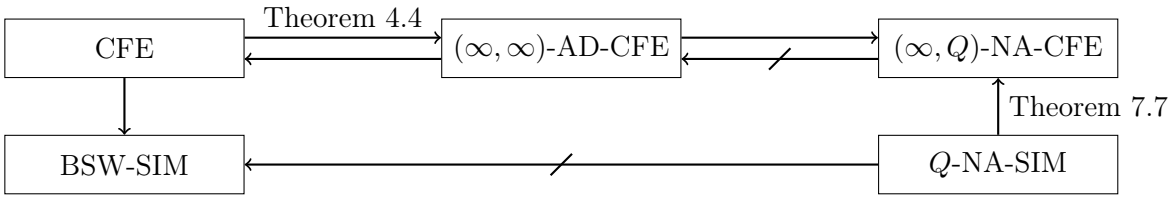


Figure 6: Implications among security definitions. BSW-SIM corresponds to [BSW11, Definition 4] and Q-NA-SIM corresponds to the non-adaptive case of [GVW12, Definition 1]. The separations follow from the impossibility of BSW-SIM and CFE-security and the possibility of Q-NA-SIM-security.

the assumed resource: For the keys that are supposed to be secret, the channel from the key authority C to the corresponding A_i has to be secure instead of only authenticated. This ensures that dishonest parties cannot learn these keys, in contrast to the (public) keys sent over only authenticated channels.

Remark. Note that the impossibility result in Section 5.1 does not rely on the fact that the encryption key is public. Hence, the impossibility extends to the case where all encryption keys are secret.

9 Conclusions and Open Problems

To better understand the space of possible security definitions for functional encryption schemes, we have put forward a new approach to defining their security. To this end, we have defined a public repository resource and repositories with access control. We then showed how a functional encryption scheme can be used to construct the latter from the former in the sense of constructive cryptography.

We have proposed CFE-security based on a security definition by Boneh, Sahai, and Waters and discussed the implications of some details in such definitions. Furthermore, we have shown that CFE-security is equivalent to constructing a repository with access control. Moreover, we have explained how impossibility results that have been known before functional encryption emerged can directly be applied in our context without reproving anything. To circumvent these impossibility results, we have considered weaker repository resources, which lead to weaker security definitions. We have further shown that a security definition for which functional encryption schemes exist is sufficient to construct such a weaker resource. Therefore, we have made explicit how those schemes can be used in a composable way. See Figure 6 for an overview of the implications among security definitions considered in this paper.

We have further outlined how to extend our constructed resource to support functions of several variables and randomized functions. Whereas traditional security definitions require non-trivial adjustments for such functions, reopening the question which definitions are appropriate, we only need to specify the involved resources and apply the definitions from the constructive cryptography framework.

An interesting goal is to find practical protocols that construct the unrestricted repository for a large set of functions using more powerful assumed resources, e.g., including a random oracle. Moreover, our constructive approach naturally leads to further open questions: To our knowledge, dishonest senders have only been considered in the context of randomized functions

and only to prevent them from tampering with the randomness. However, one could also try to find functional encryption schemes for deterministic functions that can be used to construct a repository if dishonest users can also input data. Another open problem is how several dishonest Bobs that are *not* colluding but mutually distrusting can be handled, which is a valid scenario in real applications. These problems involve more than one potentially dishonest party. While not considered in this paper, the constructive cryptography framework provides general definitions to treat such situations.

Acknowledgment

The authors would like to thank Phil Rogaway for very helpful feedback on an earlier draft of this paper. The work was supported by the Swiss National Science Foundation (SNF), project no. 200020-132794.

References

[ABF+13] Joël Alwen, Manuel Barbosa, Pooya Farshim, Rosario Gennaro, S. Dov Gordon, Stefano Tessaro, and David A. Wilson, *On the relationship between functional encryption, obfuscation, and fully homomorphic encryption*, Cryptography and Coding (Martijn Stam, ed.), Lecture Notes in Computer Science, vol. 8308, Springer Berlin Heidelberg, 2013, pp. 65–84.

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee, *Functional encryption: New perspectives and lower bounds*, Advances in Cryptology – CRYPTO 2013 (Ran Canetti and Juan A. Garay, eds.), Lecture Notes in Computer Science, vol. 8043, Springer Berlin Heidelberg, 2013, pp. 500–518.

[BCHK06] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz, *Chosen-ciphertext security from identity-based encryption*, SIAM J. Comput. **36** (2006), no. 5, 1301–1328.

[BF01] Dan Boneh and Matt Franklin, *Identity-based encryption from the Weil pairing*, Advances in Cryptology — CRYPTO 2001 (Joe Kilian, ed.), Lecture Notes in Computer Science, vol. 2139, Springer Berlin Heidelberg, 2001, pp. 213–229.

[BF13] Manuel Barbosa and Pooya Farshim, *On the semantic security of functional encryption schemes*, Public-Key Cryptography – PKC 2013 (Kaoru Kurosawa and Goichiro Hanaoka, eds.), Lecture Notes in Computer Science, vol. 7778, Springer Berlin Heidelberg, 2013, pp. 143–161.

[BO13] Mihir Bellare and Adam O'Neill, *Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition*, Cryptology and Network Security (Michel Abdalla, Cristina Nita-Rotaru, and Ricardo Dahab, eds.), Lecture Notes in Computer Science, vol. 8257, Springer International Publishing, 2013, pp. 218–234.

[BPW07] Michael Backes, Birgit Pfitzmann, and Michael Waidner, *The reactive simulatability (RSIM) framework for asynchronous systems*, Inf. Comput. **205** (2007), no. 12, 1685–1720.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters, *Functional encryption: Definitions and challenges*, Theory of Cryptography (Yuval Ishai, ed.), Lecture Notes in Computer Science, vol. 6597, Springer Berlin / Heidelberg, 2011, pp. 253–273.

[BSW12] ———, *Functional encryption: a new vision for public-key cryptography*, Commun. ACM **55** (2012), no. 11, 56–64.

[Can01] R. Canetti, *Universally composable security: a new paradigm for cryptographic protocols*, Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on, 2001, pp. 136–145.

[CFGN96] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor, *Adaptively secure multi-party computation*, Proceedings of the twenty-eighth annual ACM symposium on Theory of computing (New York, NY, USA), STOC '96, ACM, 1996, pp. 639–648.

[CIJ+13] Angelo Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano, *On the achievability of simulation-based security for functional encryption*, Advances in Cryptology – CRYPTO 2013 (Ran Canetti and Juan A. Garay, eds.), Lecture Notes in Computer Science, vol. 8043, Springer Berlin Heidelberg, 2013, pp. 519–535.

[CMT13] Sandro Coretti, Ueli Maurer, and Björn Tackmann, *Constructing confidential channels from authenticated channels—public-key encryption revisited*, Advances in Cryptology - ASIACRYPT 2013 (Kazue Sako and Palash Sarkar, eds.), Lecture Notes in Computer Science, vol. 8269, Springer Berlin Heidelberg, 2013, pp. 134–153.

[CS98] Ronald Cramer and Victor Shoup, *A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack*, Advances in Cryptology — CRYPTO '98 (Hugo Krawczyk, ed.), Lecture Notes in Computer Science, vol. 1462, Springer Berlin Heidelberg, 1998, pp. 13–25.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters, *Candidate indistinguishability obfuscation and functional encryption for all circuits*, Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on, Oct 2013, pp. 40–49.

[GGJS13] Shafi Goldwasser, Vipul Goyal, Abhishek Jain, and Amit Sahai, *Multi-input functional encryption*, Cryptology ePrint Archive, Report 2013/727, 2013.

[GJKS15] Vipul Goyal, Abhishek Jain, Venkata Koppula, and Amit Sahai, *Functional encryption for randomized functionalities*, Theory of Cryptography (Yevgeniy Dodis and Jesper Buus Nielsen, eds.), Lecture Notes in Computer Science, vol. 9015, Springer Berlin Heidelberg, 2015, pp. 325–351 (English).

[GKL+13] S. Dov Gordon, Jonathan Katz, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou, *Multi-input functional encryption*, Cryptology ePrint Archive, Report 2013/774, 2013.

[GKP+13] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich, *Reusable garbled circuits and succinct functional encryption*, Proceedings

of the 45th annual ACM symposium on Symposium on theory of computing (New York, NY, USA), STOC '13, ACM, 2013, pp. 555–564.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee, *Functional encryption with bounded collusions via multi-party computation*, Advances in Cryptology – CRYPTO 2012 (Reihaneh Safavi-Naini and Ran Canetti, eds.), Lecture Notes in Computer Science, vol. 7417, Springer Berlin Heidelberg, 2012, pp. 162–179.

[HS13] Dennis Hofheinz and Victor Shoup, *GNUC: A new universal composability framework*, Journal of Cryptology (2013), 1–86 (English).

[KRBM07] Robert König, Renato Renner, Andor Bariska, and Ueli Maurer, *Small accessible quantum information does not imply security*, Phys. Rev. Lett. **98** (2007), 140502.

[KT13] Ralf Küsters and Max Tuengerthal, *The IITM Model: a simple and expressive model for universal composability*, Cryptology ePrint Archive, Report 2013/025, 2013.

[Mau12] Ueli Maurer, *Constructive cryptography – a new paradigm for security definitions and proofs*, Theory of Security and Applications (Sebastian Mödersheim and Catuscia Palamidessi, eds.), Lecture Notes in Computer Science, vol. 6993, Springer Berlin Heidelberg, 2012, pp. 33–56.

[MR11] Ueli Maurer and Renato Renner, *Abstract cryptography*, The Second Symposium on Innovations in Computer Science, ICS 2011 (Bernard Chazelle, ed.), Tsinghua University Press, January 2011, pp. 1–21.

[MRT12] Ueli Maurer, Andreas Rüedlinger, and Björn Tackmann, *Confidentiality and integrity: A constructive perspective*, Theory of Cryptography — TCC 2012 (Ronald Cramer, ed.), Lecture Notes in Computer Science, vol. 7194, Springer Berlin Heidelberg, 2012, pp. 209–229.

[MY96] Ueli Maurer and Yacov Yacobi, *A non-interactive public-key distribution system*, Designs, Codes and Cryptography **9** (1996), no. 3, 305–316.

[Nie02] Jesper Buus Nielsen, *Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case*, Advances in Cryptology — CRYPTO 2002 (Moti Yung, ed.), Lecture Notes in Computer Science, vol. 2442, Springer Berlin Heidelberg, 2002, pp. 111–126.

[O'N10] Adam O'Neill, *Definitional issues in functional encryption*, Cryptology ePrint Archive, Report 2010/556, 2010.

[PW01] Birgit Pfitzmann and Michael Waidner, *A model for asynchronous reactive systems and its application to secure message transmission*, Proceedings of the 2001 IEEE Symposium on Security and Privacy (Washington, DC, USA), SP '01, IEEE Computer Society, 2001, pp. 184–200.

[Ren05] Renato Renner, *Security of quantum key distribution*, Ph.D. thesis, ETH Zurich, September 2005.

[Sha85] Adi Shamir, *Identity-based cryptosystems and signature schemes*, Advances in Cryptology (George Robert Blakley and David Chaum, eds.), Lecture Notes in Computer Science, vol. 196, Springer Berlin Heidelberg, 1985, pp. 47–53.

[SPPM13] Rifki Sadikin, YoungHo Park, KilHoum Park, and SangJae Moon, *Universal composability notion for functional encryption schemes*, Journal of the Korea Industrial Information System Society **18** (2013), 17–26.

[SW05] Amit Sahai and Brent Waters, *Fuzzy identity-based encryption*, Advances in Cryptology – EUROCRYPT 2005 (Ronald Cramer, ed.), Lecture Notes in Computer Science, vol. 3494, Springer Berlin Heidelberg, 2005, pp. 457–473.

A Application of the Constructed Repository Resource

In this section, we demonstrate how a repository with access control can be used in applications and how this relates to the composition theorem of the constructive cryptography framework. As a simple application, consider a university that wants its students to learn the results of their exams as well as the average result of each exam. This can be accomplished by entering the results into a repository and granting the students appropriate access rights. To formalize this, we introduce the resource $\mathsf{EXAM\_DB}$ that can directly be used by the university to enter results such that students are automatically notified about their results. We construct this resource from a repository with access control $\mathsf{REP}^{\mathrm{NA}}_{F,\infty,Q}$ as introduced in Section 7.[9] To notify the students about the results and communicate the handles needed to access the data in the repository, our construction additionally requires authenticated channels to the students.

For simplicity, we describe a resource that allows one student called Bob to access his results. As mentioned in Section 3.1, analyzing the security with a single potentially dishonest party is sufficient to guarantee security against several colluding dishonest parties, so the results here can be applied to a real world application with many students. The described protocols can be extended to such a setting in an obvious way by granting each student the appropriate rights and sending the notifications to every student.

We now describe the resource $\mathsf{EXAM\_DB}$ in more detail. After the examiner Alice inputs the ID of a lecture together with a list of students and the number of points they were awarded, Bob receives the lecture ID, the number of participants, his number of points, and the average points. More concretely, let $\mathcal{L}$ be the set of lectures and $\mathcal{S}$ be the set of students registered at the university. Now consider the resource $\mathsf{EXAM\_DB}$ that on input $(\mathsf{lecture}, ((s_1, p_1), \ldots, (s_n, p_n))) \in \mathcal{L} \times (\mathcal{S} \times \mathbb{N})^*$ at interface A, outputs $(\mathsf{lecture}, n, p, \bar{p}) \in \mathcal{L} \times \mathbb{N}^2 \times \mathbb{Q}$ at interface B, where $p = p_j$ with s_j being Bob's student ID and $\bar{p} = \frac{1}{n}\sum_{i=1}^{n} p_i$.

We will construct the resource $\mathsf{EXAM\_DB}$ from a repository with access control. To this end, let $X := \mathcal{L} \times (\mathcal{S} \times \mathbb{N})^*$. We do not require the repository to hide the length of stored entries, i.e., we set $f_0 \colon X \to \mathbb{N}, (\mathsf{lecture}, ((s_1, p_1), \ldots, (s_n, p_n))) \mapsto n$. We further consider for every student $s \in \mathcal{S}$ the function $f_s \colon X \to \mathbb{N} \cup \{\bot\}$ that maps an entry to the number of points for student s if s occurs in the list, and $\bot$ otherwise. Finally, let $f_{\mathsf{avg}} \colon X \to \mathbb{Q}, (\mathsf{lecture}, ((s_1, p_1), \ldots, (s_n, p_n))) \mapsto \frac{1}{n}\sum_{i=1}^{n} p_i$. We set $F := \{f_0, f_{\mathsf{avg}}\} \cup \{f_s \mid s \in \mathcal{S}\}$ and consider $[\mathsf{REP}^{\mathrm{NA}}_{F,\infty,Q}, \mathsf{AUT}_{A,B}]$ for $Q \geq 2|\mathcal{S}|$ (because every $s \in \mathcal{S}$ is granted access to f_{avg} and f_s) as the assumed resource in our construction.

[9] Of course, one can instead use REP_F if it is available, since $\mathsf{REP}^{\mathrm{NA}}_{F,\infty,Q}$ is more restricted than REP_F.

The protocol $\pi^{\mathrm{Ex}} := (\pi_A^{\mathrm{Ex}}, \pi_B^{\mathrm{Ex}}, \pi_C^{\mathrm{Ex}})$ is defined as follows: Initially, π_C^{Ex} outputs f_{Bob} and f_{avg} at its inner interface. The converter π_A^{Ex} on input $x = (\mathsf{lecture}, ((s_1, p_1), \ldots, (s_n, p_n))) \in \mathcal{L} \times (\mathcal{S} \times \mathbb{N})^*$ at its outer interface, outputs x at its inner interface to the repository. When the repository returns a handle h, π_A^{Ex} sends $(\mathsf{lecture}, h)$ to Bob over the authenticated channel. On input $(\mathsf{lecture}, h)$ at its inner interface, π_B^{Ex} inputs (f_0, h), (f_{Bob}, h), and (f_{avg}, h) at its inner interface to the repository. Let the returned values from the repository be n, p, and $\bar{p}$, respectively. The converter then outputs $(\mathsf{lecture}, n, p, \bar{p})$ at its outer interface. We now show that this protocol indeed constructs the desired resource $\mathsf{EXAM\_DB}$.

Proposition A.1. *We have*

$$[\mathsf{REP}_{F,\infty,Q}^{\mathrm{NA}}, \mathsf{AUT}_{A,B}] \overset{\pi^{\mathrm{Ex}}}{\Longrightarrow} \mathsf{EXAM\_DB}.$$

Proof. It is easy to see that

$$\pi_A^{\mathrm{Ex}} \pi_B^{\mathrm{Ex}} \pi_C^{\mathrm{Ex}} [\mathsf{REP}_{F,\infty,Q}^{\mathrm{NA}}, \mathsf{AUT}_{A,B}] \equiv \mathsf{EXAM\_DB}.$$

We define a simulator σ_B as follows: Initially, it outputs f_{Bob} and f_{avg} at its outer sub-interface simulating $\mathsf{REP}_{F,\infty,Q}^{\mathrm{NA}}$. On input $(\mathsf{lecture}, n, p, \bar{p})$ at its inner interface, it invokes $h \leftarrow \texttt{getHandle}$, internally stores $(h, n, p, \bar{p})$, and outputs $(\mathsf{lecture}, h)$ at its outer sub-interface simulating $\mathsf{AUT}_{A,B}$. On input (f_0, h), (f_{Bob}, h), or (f_{avg}, h) at its outer interface for some h that has been stored, it outputs the corresponding stored n, p, or $\bar{p}$, respectively, at its outer sub-interface simulating $\mathsf{REP}_{F,\infty,Q}^{\mathrm{NA}}$. Other inputs are ignored. We then have

$$\pi_A^{\mathrm{Ex}} \pi_C^{\mathrm{Ex}} [\mathsf{REP}_{F,\infty,Q}^{\mathrm{NA}}, \mathsf{AUT}_{A,B}] \equiv \sigma_B \mathsf{EXAM\_DB}$$

and the claim follows. □

By the composition theorem mentioned in Section 2.3, one can first construct $\mathsf{REP}_{F,\infty,Q}^{\mathrm{NA}}$ from a public repository via a protocol π' as demonstrated in Section 7 and then use π^{Ex} to construct $\mathsf{EXAM\_DB}$. Due to this modularity, the protocol π^{Ex} does not need to consider functional encryption or know how $\mathsf{REP}_{F,\infty,Q}^{\mathrm{NA}}$ was constructed. This makes the protocol and its security analysis very simple. The composition theorem guarantees that the overall construction (corresponding to an application of the protocol $\pi^{\mathrm{Ex}} \circ \pi'$) is secure. Moreover, security is still guaranteed if $\mathsf{AUT}_{A,B}$ is in parallel constructed from an insecure channel using some secure authentication protocol. Traditional security definitions for functional encryption do not offer such guarantees.

Note that π_C^{Ex} has to grant Bob the rights before A makes any inputs. Using the constructive approach, this is obvious since $\mathsf{REP}_{F,\infty,Q}^{\mathrm{NA}}$ does not allow any inputs at interface C after an input at interface A. In a real world application, this corresponds to sending all keys to the students before the exam session starts (and using fresh keys every semester). If a student forgot to register and asks for his key after some results have already been published, the university cannot give him a key without destroying all security guarantees. When a functional encryption scheme is used to build $\mathsf{EXAM\_DB}$ from scratch using traditional security definitions, this fact might be less obvious.