A Constructive Approach to Functional Encryption

Christian Matt
ETH Zurich
mattc@inf.ethz.ch

Ueli Maurer
ETH Zurich
maurer@inf.ethz.ch

Abstract

Functional encryption is an important generalization of several types of encryption such as public-key, identity-based, and attribute-based encryption. Numerous different security definitions for functional encryption have been proposed, most of them being rather complex and involving several algorithms. Many of these definitions differ in details such as which algorithm has oracle access to which oracle, while the consequences of specific choices are often unclear. This spans a large space of possible definitions without a consensus on the adequacy of specific points in this space. What a particular definition means and for which applications it is suitable remains unsettled.

To remedy this situation, we propose a novel interpretation of functional encryption, based on the Constructive Cryptography framework, in which a protocol is seen as a construction of an ideal resource with desired properties from a real resource, which is assumed to be available. The resulting ideal resource can then be used as a real resource in other protocols to construct more advanced resources. The real resource we consider here corresponds to a public repository that allows everyone to read its contents. Such repositories are indeed widely available on the internet. Using functional encryption, we construct, as the ideal resource, a repository with fine-grained access control.

Based on this constructive viewpoint, we propose a new security definition, called FA-security, for functional encryption by adequately modifying an established definition, and prove the equivalence to our notion of construction. This gives evidence that FA-security is an appropriate definition. We further consider known impossibility results and examine a weaker security definition. We show that this weaker definition, for which secure schemes exist, is sufficient to construct a repository that restricts the number and order of interactions. This makes explicit how such schemes can be used.

1 Introduction

Consider a public repository that allows users to input and access data. When users input a piece of data, they get a handle via which this data can be accessed by other users. Such repositories are commonly available in real applications. For example, one can upload documents to a web server, which corresponds to the repository, and these documents can then be accessed by everyone who knows the URL, which corresponds to the handle. A relevant goal is to add access control to such a repository. That is, instead of allowing everyone to access all data, we want to be able to grant users access rights and prevent users without the required rights from accessing the data. In contrast to many systems used in practice, including most cloud services, we want to achieve this without trusting the repository provider. To this end, we consider a trusted party that is external to the repository and that can grant users access rights. As outlined below, this also allows us to model as a special case that the users who upload data specify access rights for their data.

One can use encryption to restrict access to the data: Using a public-key encryption scheme, the trusted party can generate a key-pair, publish the public key and give the private key to the users who should be able to access the data. Users can then input encrypted data and exactly those who were given the secret key can access it. However, this does not allow much flexibility; one can only specify whether a user is allowed to access all data or nothing. One might prefer to grant different users rights to access different information about the data. This is indeed possible if an encryption scheme with more features is used. Identity-based encryption [Sha85, MY96, BF01], for example, 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. This therefore enables the user who inputs data to specify who

should be able to access it. More advanced access policies can be implemented using attribute-based encryption [SW05].

Functional encryption 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, and attribute-based encryption can be seen as a special case of functional encryption. Briefly, a functional encryption scheme for a set of functions allows a trusted authority holding a master secret key to generate secret keys for all functions in this set. Given a secret key for a function f and an encryption of a value x, one can efficiently compute $f(x)$ but does not learn anything more about x.

Encrypting data in a repository with a functional encryption scheme allows to grant users the right to access certain functions of the input data. The more functions the scheme supports, the more flexibility the resulting repository provides. The following application from [BSW12, GKP+13] demonstrates the benefits of such flexibility: 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.

Formalizing the intuitive security requirement that one only learns $f(x)$ given a ciphertext for some x and a secret key for f has caused more trouble than one might expect; 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 [BSW11, AGVW13, BO12].

To address the question which definition is the right one, we use the Constructive Cryptography framework [Mau12, MR11], in which a protocol is seen as a construction of a so-called ideal resource from a so-called real resource. This ideal resource can then be used in a larger protocol as a real resource to construct a more advanced ideal resource. The security of the overall construction follows due to a composition theorem of the framework. This allows us to capture the security of a protocol by what it achieves, namely which resource it constructs. Other frameworks that capture security properties by defining an ideal functionality include Universal Composability [Can01] and Reactive Simulatability [PW01, BPW07]. While these frameworks are designed bottom-up from a specific machine model, the Constructive Cryptography framework follows a top-down approach, leading to simpler descriptions and avoiding technicalities.

Defining security in a composable framework has the advantage that all security properties are guaranteed to hold in any context. As an example, consider a company that uses many different repositories to store related data where employees have different rights for each repository. While it might not be clear whether a protocol proven to satisfy a standalone security definition is secure in such a context or a new definition is required that explicitly takes this application into account, our approach makes sure that all users only learn what can be concluded from the information they have access rights to in the repositories.

We define a real resource corresponding to a public repository without access control and an ideal resource corresponding to a repository that allows users to access certain functions of the input data. For each function in the set of supported functions, a trusted party can grant the right to access this function of all stored data. We then define a protocol that uses a functional encryption scheme in a natural way to construct this ideal resource from the real resource. The underlying functional encryption scheme can be considered secure if this construction is achieved. From this perspective, we propose a new security definition for functional encryption schemes, which we call fully adaptive security (FA-security), based on a definition from [BSW11] and show that our protocol securely constructs the ideal resource if and only if the underlying functional encryption scheme is FA-secure. Since known impossibility results extend to FA-security, we also consider a weaker definition from [GVW12] and show that it is sufficient to construct a restricted repository with access control. This justifies that definition and makes explicit for which applications schemes satisfying it can be used.

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.

Two types of systems we consider here are *resources* and *converters*. Resources are denoted by small capitals and have a finite set $\mathcal{I}$ of interfaces. 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 R. The outer interface of α then serves as the new interface I of the composed resource, which is denoted by $\alpha^I\mathrm{R}$. We also write $\alpha_I\mathrm{R}$ instead of $\alpha_I^I\mathrm{R}$ for a converter α_I. For $\mathcal{I}$-resources $\mathrm{R}_1, \ldots, \mathrm{R}_m$, the *parallel composition* $[\mathrm{R}_1, \ldots, \mathrm{R}_m]$ is defined as the $\mathcal{I}$-resource where each interface $I \in \mathcal{I}$ allows to access the corresponding interfaces of all sub-systems R_i as sub-interfaces.

A *distinguisher* 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 $\mathrm{P}(D\mathrm{R} = 1)$ to denote the probability that D outputs the bit 1 when connected to resource 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 D for resources R and S is defined as

$$\Delta^D(\mathrm{R}, \mathrm{S}) := |\mathrm{P}(D\mathrm{R} = 1) - \mathrm{P}(D\mathrm{S} = 1)|\,.$$

If $\Delta^D(\mathrm{R}, \mathrm{S}) = 0$ for all distinguishers D, we say R and S are *equivalent*, denoted as $\mathrm{R} \equiv \mathrm{S}$. If the distinguishing advantage is negligible for all efficient distinguishers, we say R and S are *computationally indistinguishable*, denoted as $\mathrm{R} \approx \mathrm{S}$.

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. If a channel is used in a context with several potentially dishonest parties, all of them are connected to interface E. 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 as $\textsc{Aut}_{A,B}$, 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. Other inputs are ignored.

This channel is called *authenticated* because E cannot modify the messages. If an eavesdropper can only learn the length of the transferred messages, we get the following resource.

Definition 2.3. A *secure channel from A to B*, denoted as $\textsc{Sec}_{A,B}$, 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. Other inputs are ignored.

2.3 Construction of Resources

A *protocol* is a vector of converters with the purpose of constructing a so-called ideal resource from an available real resource. 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 $\textsc{Sec}_{A,B}$ from authenticated channels $\textsc{Aut}_{B,A}$ and $\textsc{Aut}_{A,B}$ in presence of a dishonest eavesdropper E. Here, the real resource is $\mathrm{R} := [\textsc{Aut}_{B,A}, \textsc{Aut}_{A,B}]$ and the

ideal resource is $\mathrm{S} := \textsc{Sec}_{A,B}$. In such a setting, a protocol $\pi = (\pi_A, \pi_B)$ constructs S from R with potentially dishonest E if there exists a converter σ_E (called *simulator*) such that

$$\begin{aligned} \pi_A\pi_B\bot^E\mathrm{R} &\approx \bot^E\mathrm{S} \\ \text{and} \qquad \pi_A\pi_B\mathrm{R} &\approx \sigma_E\mathrm{S}, \end{aligned}$$

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

While Eve in the above example can be seen as an attacker who is not guaranteed any interaction, the setting in this paper includes one potentially dishonest party that can also be honest. Hence, all parties will have a protocol and all interactions provided by the ideal resource are guaranteed to all honest parties. On the other hand, a dishonest party should not be able to do more than specified by the ideal resource. We will consider three parties A, B, and C, where B is potentially dishonest and define a secure construction as follows.

Definition 2.4. Let R and S be $\{A, B, C\}$-resources and let $\pi = (\pi_A, \pi_B, \pi_C)$ be a protocol. We say π *constructs* S from R *with potentially dishonest* B if there exists a converter σ_B such that

$$\begin{aligned} \pi_A\pi_B\pi_C\mathrm{R} &\approx \quad \mathrm{S} \\ \text{and} \qquad \pi_A\pi_C\mathrm{R} &\approx \sigma_B\mathrm{S}. \end{aligned}$$

This definition is a special case of the abstraction notion from [MR11] that considers many dishonest and mutually distrusting parties.

2.4 Efficiency and Security Parameters

Cryptographic primitives are often equipped with a security parameter and efficiency and negligibility is defined with respect to this parameter. To simplify the presentation, we will omit security parameters in this work. To be compatible with standard asymptotic definitions, one can understand all results in this paper asymptotically by treating all algorithms and systems as asymptotic families indexed by a security parameter. The distinguishing advantage is then a function of this parameter. All reductions in this paper are efficient with respect to standard polynomial-time notions.

2.5 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 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.

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 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 $\mathtt{sk}_f$ for this f where $\mathtt{sk}_{f_0}$ equals the empty string. Given `pk` and some $x \in X$, `enc` computes a ciphertext c such that $\mathtt{dec}(\mathtt{sk}_f, c) \neq f(x)$ with negligible probability, 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.

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 $\textsc{Rep}_F$ 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.[1]

Remark. Our definitions can be straightforwardly generalized to allow probabilistic access functions where the randomness is provided by the repository and access functions that depend on more than one piece of data, i.e., functions with domain X^n for arbitrary $n \in \mathbb{N}$. Since we do not need this generality for the examples discussed in this paper, we only consider the simplest case.

We now define a public repository without access control, which will serve as a real resource in our constructions. It corresponds to a repository as defined above where everyone is allowed to access the identity function of stored data.

Definition 3.2. Let X be a nonempty set, $f_0 := \mathrm{id}_X : X \to X, x \mapsto x$, and $P := \{f_0\}$. We define the *public repository for* X as $\mathrm{PREP}_X := \mathrm{REP}_P$. 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 $\mathcal{E} = (\mathtt{setup}, \mathtt{keygen}, \mathtt{enc}, \mathtt{dec})$ be a functional encryption scheme for a set F of functions with domain X and let $\mathfrak{C}$ be the range of $\mathtt{enc}$. Our goal is to construct REP_F from $\mathrm{PREP}_{\mathfrak{C}}$. To distribute keys in the real world, we additionally need an authenticated channel $\mathrm{AUT}_{C,A}$ from Charlie to Alice and a secure channel $\mathrm{SEC}_{C,B}$ from Charlie to Bob[2], i.e., the real resource in our construction corresponds to $[\mathrm{PREP}_{\mathfrak{C}}, \mathrm{AUT}_{C,A}, \mathrm{SEC}_{C,B}]$.

The protocol $\pi = (\pi_A, \pi_B, \pi_C)$ works as follows: At the beginning, π_C invokes $(\mathtt{pk}, \mathtt{mk}) \leftarrow \mathtt{setup}()$, stores $\mathtt{mk}$ and sends $\mathtt{pk}$ 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 $c := \mathtt{enc}(\mathtt{pk}, x)$ 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 sends $(f, \mathtt{keygen}(\mathtt{mk}, f))$ 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 $\mathtt{sk}_f$ for this function f or if $f = f_0$. If it receives a ciphertext c from the repository, it outputs $\mathtt{dec}(\mathtt{sk}_f, c)$ at its outer interface. All other inputs are ignored.

The following lemma states that this protocol constructs the desired ideal 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*

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

4 Security of Functional Encryption Schemes

4.1 Definition of FA-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 *fully adaptive security* (*FA-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 follow the notation from [BSW11], i.e., for algorithms A

[1] 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 here.

[2] 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.

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 the value of s.

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

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

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

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

$(\texttt{pk}, s) \leftarrow S_1()$
$(l, \tau) \leftarrow (0, 0)$
repeat
  $l \leftarrow l + 1$
  $x_l \leftarrow \mathrm{Adv}_1^{f \mapsto S_2(f, f(x_1), \ldots, f(x_{l-1}))[[s]]}(\texttt{pk})[[\tau]]$
  $(f_1, \ldots, f_q) \leftarrow$ 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 $t = \texttt{true}$
return τ

We say $\mathcal{E}$ is *FA-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.

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 round. 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 ideal resource that is constructed.

4.2 Equivalence of FA-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 FA-secure. This implies that FA-security is precisely the definition needed for our purpose. The following lemma shows that FA-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 D for $\pi_A\pi_C[\text{PREP}_{\mathfrak{C}}, \text{AUT}_{C,A}, \text{SEC}_{C,B}]$ and $\sigma_B\text{REP}_F$, there is an efficient probabilistic oracle algorithm Adv_1, an efficient probabilistic algorithm Adv_2, and an efficient distinguisher D' for the FA experiment such that*

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

Proof. We define σ_B as follows:

Initialization

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

Inner Interface

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

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 $\text{PREP}_{\mathfrak{C}}$
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**
$y_i \leftarrow$ returned value from output (f_i, h) at inner interface
$c_l \leftarrow S_3(y_0, \ldots, y_q)[[s]]$
output c_l at outer sub-interface simulating $\text{PREP}_{\mathfrak{C}}$

Now let D be an efficient distinguisher for $\pi_A\pi_C[\text{PREP}_{\mathfrak{C}}, \text{AUT}_{C,A}, \text{SEC}_{C,B}]$ and $\sigma_B\text{REP}_F$. We can assume without loss of generality that 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 h.

We now describe the algorithms Adv_1 and Adv_2. When Adv_1 is invoked with $\mathtt{pk}$ and $\tau = 0$, it starts a new simulation of the distinguisher D, outputting $\mathtt{pk}$ at interface B from the authenticated channel. When D returns a bit b, Adv_1 sets $\tau \leftarrow (\mathtt{return}, b)$ and returns a random $x \in X$. When D inputs $f \in F$ at interface C, Adv_1 invokes its oracle with query f and outputs f and the answer to D at interface B from the secure channel. When 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 D inputs $h \in H$ at interface B, Adv_1 saves M and the state of D in τ and returns $M[h]$. This will invoke Adv_2 on input a ciphertext c and τ. If $\tau = (\mathtt{return}, b)$, 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 $\mathtt{pk}$ and $\tau \neq 0$. In this case, it reads c and the state of D from τ and continues the simulation by outputting c at interface B from the repository. Adv_1 then proceeds as above.

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

The next lemma shows that FA-security is not only sufficient but also necessary for constructions of the desired repository resource. Since this notion is even stronger than the one in [BSW11], 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* D *for the FA experiment, there exists an efficient distinguisher* D' *for* $\pi_A\pi_C[\mathrm{PREP}_{\mathfrak{C}}, \mathrm{AUT}_{C,A}, \mathrm{SEC}_{C,B}]$ *and* $\sigma_B\mathrm{REP}_F$ *such that*

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

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 $(f, \mathtt{sk}_f)$ at its outer interface, S_2 stores the state of σ_B together with M, q, and f_q in s and returns $\mathtt{sk}_f$. 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[3] $h_l \leftarrow \mathtt{getHandle}$, 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 to 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 D be an efficient distinguisher for the FA experiment. We define a distinguisher D' for the resources $\pi_A\pi_C[\mathrm{PREP}_{\mathfrak{C}}, \mathrm{AUT}_{C,A}, \mathrm{SEC}_{C,B}]$ and $\sigma_B\mathrm{REP}_F$ as follows. It first runs Adv_1 on input the initial output $\mathtt{pk}$ 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 $(f, \mathtt{sk}_f)$ at interface B, and returning $\mathtt{sk}_f$ as the answer. When Adv_1 returns x, D' inputs x at interface A and then the returned handle[4] h at interface B to the repository. When c is output at interface B, the distinguisher D' invokes Adv_2 on input c and τ. If it returns $t = \mathtt{false}$, D' repeats this procedure by running Adv_1 on input $\mathtt{pk}$ and τ. Otherwise, D' invokes D on input τ. Finally, D' outputs the output of D. Since the distribution of τ if D' is connected to $\pi_A\pi_C[\mathrm{PREP}_{\mathfrak{C}}, \mathrm{AUT}_{C,A}, \mathrm{SEC}_{C,B}]$ is the same as in the real FA experiment and the same as in the ideal one if D' is connected to $\sigma_B\mathrm{REP}_F$, the corresponding distinguishing advantages are equal. □

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

Theorem 4.4. *The protocol* π *defined above constructs* REP_F *from* $[\mathrm{PREP}_{\mathfrak{C}}, \mathrm{AUT}_{C,A}, \mathrm{SEC}_{C,B}]$ *with potentially dishonest* B *if and only if* $\mathcal{E}$ *is FA-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 the security definition from [BSW11] 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 $\mathcal{E} = (\mathtt{setup}, \mathtt{keygen}, \mathtt{enc}, \mathtt{dec})$ 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 $\mathcal{E}' = (\mathtt{setup}', \mathtt{keygen}', \mathtt{enc}, \mathtt{dec}')$ as follows. The algorithm $\mathtt{setup}'$ first runs $\mathtt{setup}$ 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 $\mathtt{pk}$ (and discards the trapdoor). The algorithm $\mathtt{keygen}'$ on input $(\mathtt{mk}, f)$ does the same as $\mathtt{keygen}$ if $f \neq p^*$ and returns $(p^*, \mathtt{sk}_{\mathrm{id}})$ with $\mathtt{sk}_{\mathrm{id}} \leftarrow \mathtt{keygen}(\mathtt{mk}, \mathrm{id}_X)$ if $f = p^*$. The algorithm $\mathtt{enc}$ is not modified and $\mathtt{dec}'$ on input $((p^*, \mathtt{sk}_{\mathrm{id}}), c)$ returns $p^*(\mathtt{dec}(\mathtt{sk}_{\mathrm{id}}, c))$ and $\mathtt{dec}(\mathtt{sk}_f, c)$ on input $(\mathtt{sk}_f, c)$. As in [BF13], it can be shown that $\mathcal{E}'$ is FA-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 generated the public key. We claim that this is actually not a problem: An adversary cannot choose for

[3] If $\mathtt{getHandle}$ requires interaction with interface A, S_3 emulates it using an arbitrary fixed strategy.

[4] D' uses the same strategy as S_3 for inputs at interface A for $\mathtt{getHandle}$, such that handles are equally distributed.

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 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. Using this permutation in another protocol built on top of the modified scheme would be problematic. However, if schemes are composed according to the Constructive Cryptography framework, this cannot happen since other protocols then only use the ideal resource that is constructed by the scheme and our ideal resource does not provide a distinguished function corresponding to the permutation in the public key to any party.

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 $X := M$ and $F_{\mathrm{PKE}} := \{f_0, \mathrm{id}_X\}$ with $f_0 \colon X \to \mathbb{N}, x \mapsto |x|$. 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 ideal repository resource defined above, 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, BO12] 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 [BO12].

Theorem 5.1. *There is no FA-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 FA-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 real resource or aim for a weaker ideal 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 real resource. Bellare and O'Neill [BO12] 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 ideal resource.

Many papers consider different security definitions [O'N10, BO12, GVW12, AGVW13, GKP+13]. 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. In the following section, we further follow the approach of considering weaker ideal resources. In particular, we introduce restricted repositories and show that schemes satisfying a definition from [GVW12] can be used to construct such repositories.

6 Constructing Restricted Versions of Repository Resources

6.1 Definitions

We first define restricted variants of repository resources.

Definition 6.1. 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 $\text{REP}^{\text{AD}}_{F,L,Q}$ 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., $\text{REP}^{\text{AD}}_{F,\infty,\infty} \equiv \text{REP}_F$). We further define a nonadaptive variant $\text{REP}^{\text{NA}}_{F,L,Q}$ 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 $\mathcal{E} = (\texttt{setup}, \texttt{keygen}, \texttt{enc}, \texttt{dec})$ for a set F of functions with domain X where $\mathfrak{C}$ is the range of `enc`. We then define the protocol $\pi^{\text{AD},L,Q} := (\pi^L_A, \pi_B, \pi^{\text{AD},Q}_C)$ as π in section 3.2, but π^L_A and $\pi^{\text{AD},Q}_C$ 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 constructs a restricted repository with access control if all parties are honest. It follows directly from the correctness of the functional encryption scheme.

Lemma 6.2. *For the protocol $\pi^{\text{AD},L,Q} = (\pi^L_A, \pi_B, \pi^{\text{AD},Q}_C)$ defined above, we have*

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

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 $\text{PREP}^{\emptyset}_X$ that is identical to PREP_X as in Definition 3.2 but additionally accepts the input `isEmpty` at interface C which is answered with `true` if the repository is empty, and `false` otherwise. We then define $\pi^{\text{NA},L,Q} := (\pi^L_A, \pi_B, \pi^{\text{NA},Q}_C)$ where $\pi^{\text{NA},Q}_C$ on each input at its outer interface outputs `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 $\pi^{\text{AD},Q}_C$.

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

Lemma 6.3. *For the protocol $\pi^{\text{NA},L,Q} = (\pi^L_A, \pi_B, \pi^{\text{NA},Q}_C)$ defined above, we have*

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

We now recall a simulation-based single-message security definition from [GVW12].

Definition 6.4. Let $\mathcal{E} = (\texttt{setup}, \texttt{keygen}, \texttt{enc}, \texttt{dec})$ be a functional encryption scheme for a set F of functions with domain X. We introduce the following experiments for an efficient probabilistic oracle algorithm Adv_1 and efficient probabilistic algorithms Adv_2 and S. We denote the advantage of a distinguisher D in distinguishing the outputs of these experiments by $\Delta^D\left(\text{NA-SIM-Exp}^{\text{Real}}_{\mathcal{E},\text{Adv}}, \text{NA-SIM-Exp}^{\text{Ideal}}_{\mathcal{E},\text{Adv},S}\right)$.

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

For $Q \in \mathbb{N}$, the scheme $\mathcal{E}$ is *Q-NA-SIM-secure* if there exists an efficient probabilistic algorithm S such that $\Delta^D\left(\text{NA-SIM-Exp}^{\text{Real}}_{\mathcal{E},\text{Adv}}, \text{NA-SIM-Exp}^{\text{Ideal}}_{\mathcal{E},\text{Adv},S}\right)$ 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 D.

6.2 Sufficiency of the Definition

The following result implies that the above definition is sufficient to construct a nonadaptive repository with potentially dishonest B. 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.

Theorem 6.5. *Let $L,Q \in \mathbb{N}$ and S be an efficient probabilistic algorithm. Then there exists an efficient converter σ_B such that for all efficient distinguishers D for $\pi_A^L\pi_C^{\mathrm{NA},Q}[\mathrm{PREP}^{\emptyset}_{\mathfrak{C}}, \mathrm{AUT}_{C,A}, \mathrm{SEC}_{C,B}]$ and $\sigma_B\mathrm{REP}^{\mathrm{NA}}_{F,L,Q}$, 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 D' for the NA-SIM experiment such that*

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

Proof. We define σ_B as follows:

Initialization

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

Inner Interface

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

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 $\text{PREP}^{\emptyset}_{\mathfrak{C}}$
  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**
      $y_i \leftarrow$ returned value from output (f_i, h) at inner interface
    $c_l \leftarrow S(\mathtt{pk}, f_1, \ldots, f_q, \mathtt{sk}_1, \ldots \mathtt{sk}_q, y_0, \ldots, y_q)$
    output c_l at outer sub-interface simulating $\text{PREP}^{\emptyset}_{\mathfrak{C}}$

Now let D be an efficient distinguisher for the two resources $\pi^L_A \pi^{\text{NA},Q}_C[\text{PREP}^{\emptyset}_{\mathfrak{C}}, \text{AUT}_{C,A}, \text{SEC}_{C,B}]$ and $\sigma_B \text{REP}^{\text{NA}}_{F,L,Q}$. We can assume that 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, 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 D' emulate D. At the beginning, Adv_1 sets $l \leftarrow 0$, draws a number $\hat{l} \in \{1, \ldots, L\}$ uniformly at random and outputs $\mathtt{pk}$ at interface B from the authenticated channel for D. When D inputs $f \in F$ at interface C, Adv_1 makes the oracle-query f and outputs f and the answer $\mathtt{sk}$ at interface B from the secure channel. When D inputs x at interface A, Adv_1 invokes $\mathtt{getHandle}$ and outputs the returned handle h at interface A. It further sets $M[h] \leftarrow x$ for a map M. When D inputs $h \in H$ at interface B, Adv_1 increments l by one. If $l < \hat{l}$, Adv_1 outputs $\mathtt{enc}(\mathtt{pk}, M[h])$ 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 $\mathtt{sk}_1, \ldots, \mathtt{sk}_q$, and the state of D in τ and returns $(M[h], \tau)$. On input $(\mathtt{pk}, c, \tau)$, Adv_2 reads M, $f_1, \ldots, f_q$, $\mathtt{sk}_1, \ldots, \mathtt{sk}_q$, and the state of D from τ and continues the simulation of D by outputting c at interface B from the repository. When D inputs x at interface A, Adv_2 invokes $\mathtt{getHandle}$, outputs the returned handle h at interface A, and sets $M[h] \leftarrow x$. When 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 $c' \leftarrow S(\mathtt{pk}, f_1, \ldots, f_q, \mathtt{sk}_1, \ldots \mathtt{sk}_q, y_0, \ldots, y_q)$ and outputs c' at interface B from the repository. When D outputs a bit, Adv_2 returns this bit. The distinguisher 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 H_i that corresponds to $\pi^L_A \pi^{\text{NA},Q}_C[\text{PREP}^{\emptyset}_{\mathfrak{C}}, \text{AUT}_{C,A}, \text{SEC}_{C,B}]$ until i different $h \in H$ are input at interface B, and from then on outputs $S(\mathtt{pk}, f_1, \ldots, f_q, \mathtt{sk}_1, \ldots \mathtt{sk}_q, y_0, \ldots, y_q)$ at interface B on input $h \in H$, where q is the number of inputs at interface C, f_j corresponds to the jth input at interface C, $\mathtt{sk}_j$ 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. Note that $\text{H}_L \equiv \pi^L_A \pi^{\text{NA},Q}_C[\text{PREP}^{\emptyset}_{\mathfrak{C}}, \text{AUT}_{C,A}, \text{SEC}_{C,B}]$ and $\text{H}_0 \equiv \sigma_B \text{REP}^{\text{NA}}_{F,L,Q}$. Further note that

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

and similarly

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

We therefore have

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

7 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 define 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 a repository with access control from a public repository in the sense of the Constructive Cryptography framework.

We have proposed a traditional, experiment-based security definition for functional encryption based on a definition by Boneh, Sahai, and Waters and discussed the implications of some details in such definitions. Furthermore, we have shown that this definition is equivalent to securely constructing an ideal repository with access control and no restrictions on the inputs. Moreover, we have explained how known impossibility results can be applied to our definition. We have further considered another experiment-based definition for which functional encryption schemes exist and have shown that this definition is sufficient to construct a repository with access control that only allows to grant rights before any data is input, making explicit how such schemes can be used.

An interesting goal is to find practical protocols that construct the unrestricted ideal repository for a large set of functions using stronger real resources such as random oracles. Moreover, our constructive approach naturally leads to further open questions, which, to our knowledge, have not been considered so far: We have only considered constructions with an honest Alice because it seems that existing definition in the literature try to capture this setting. However, one could try to find functional encryption schemes that can be used to construct an ideal repository if dishonest users can also input data. This probably leads to stronger security definitions in the spirit of CCA-security instead of CPA-security for public-key encryption. 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.

References

[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 JuanA. Garay, eds.), Lecture Notes in Computer Science, vol. 8043, Springer Berlin Heidelberg, 2013, pp. 500–518.

[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 (English).

[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.

[BO12] Mihir Bellare and Adam O'Neill, *Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition*, Cryptology ePrint Archive, Report 2012/515, 2012.

[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.

[CMT13] Sandro Coretti, Ueli Maurer, and Björn Tackmann, *Constructing confidential channels from authenticated channels—public-key encryption revisited*, Advances in Cryptology - ASIACRYPT 2013, Lecture Notes in Computer Science, Dec 2013, To appear.

[DH76] Whitfield Diffie and Martin E. Hellman, *New directions in cryptography*, Information Theory, IEEE Transactions on **22** (1976), no. 6, 644–654.

[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.

[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 in 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 (English).

[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.

[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.

[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.