Token-Based Multi-input Functional Encryption

Nuttapong Attrapadung[1], Goichiro Hanaoka[1], Takato Hirano[2], Yutaka Kawai[2(✉)], Yoshihiro Koseki[2], and Jacob C. N. Schuldt[1]

[1] National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
[2] Mitsubishi Electric Corporation, Tokyo, Japan
Kawai.Yutaka@da.MitsubishiElectric.co.jp

Abstract. In this paper, we put forward the notion of a token-based multi-input functional encryption (token-based MIFE) scheme – a notion intended to give encryptors a mechanism to control the decryption of encrypted messages, by extending the encryption and decryption algorithms to additionally use tokens. The basic idea is that a decryptor must hold an appropriate decryption token in addition to his secrete key, to be able to decrypt. This type of scheme can address security concerns potentially arising in applications of functional encryption aimed at addressing the problem of privacy preserving data analysis. We firstly formalize token-based MIFE, and then provide two basic schemes based on an ordinary MIFE scheme and a public key encryption scheme and a pseudorandom function (PRF), respectively. Lastly, we extend the latter construction to allow decryption tokens to be restricted to specified set of encryptions, even if all encryptions have been done using the same encryption token. This is achieved by using a constrained PRF.

1 Introduction

1.1 Background and Motivation

Nowadays, large amounts of data is constantly being collected, and data analysis has become an indispensable tool for extracting value from this data. However, central data collection and processing, which is typically at the heart of the data collection, potentially leads to security or privacy issues, as the storage and processing provider, such as a cloud environment, is often not fully trusted to keep the data or the extracted information private. This is especially a concern, if the collected data contains sensitive information. As a potential solution, homomorphic encryption (HE) and functional encryption (FE) have attracted attention. Especially, multi-key homomorphic encryption (MKHE) or multi-input functional encryption (MIFE) is expected to be suitable for the case where the data is collected by different entities and will be processed by an entity which is not fully trusted. Recently, MIFE for inner products has been studied [BLR+14, ARW16, LL16, KLM+16, ABDP15, DOT18] since the inner product operation frequently appears in various statistical computation.

J. Baek et al. (Eds.): ProvSec 2018, LNCS 11192, pp. 147–164, 2018.
https://doi.org/10.1007/978-3-030-01446-9_9

Let us consider the following data analysis scenario using HE or FE. A data analyst wants to analyze users' data stored on a cloud server, but is only trusted with the result of the analysis, and not the individual data of the users. In order to achieve this securely, users might encrypt their own data by using the HE or FE public key of the data analyst, and the store the ciphertexts in the cloud. Here, we refer to each user's ciphertext stored in the cloud as the *original ciphertext*. In the HE case, the data analyst might request the cloud to perform homomorphic evaluations of the original ciphertexts corresponding to the desired analysis, and will then be able to obtain the result by decrypting the evaluated ciphertext using his own (master) secret key. In the FE case, the data analyst can obtain the result of the data analysis by merely decrypting the original ciphertext(s) under his own secret key which embeds a function corresponding to the desired analysis.

The above approach might seem to be a secure way for the analyst to obtain the desired result. However, this might not be the case in all scenarios. For example, in the HE case, the analyst might be able to instruct the cloud environment to do a different type of processing or to limit the data which is being processed such that additional details regarding the data of individual users are leaked. In the extreme case, the analyst gains access to the original ciphertexts, in which case he can directly obtain the data of the individual users. In the FE case, analysts will often be required to have access to many different keys implementing various functions, and decrypting the user data with all of these will potentially leak unintended information regarding the user data. Furthermore, and perhaps more importantly, a different analysts holding a key corresponding to a different function might gain access the original ciphertexts and decrypt these using his key. As the data was intended for the original analyst holding a key for the original function, this might lead to unintended data leaks. This problem might be amplified if the key of one data analyst is compromised, as this will put all existing and future data at risk. Hence, in these scenarios, additional security measures might be warranted.

1.2 Our Contributions

In this paper, we attempt to address the above described problem, and focus on reducing the power of the master or user secret key. We propose a new encryption primitive, called *token-based* encryption, which provides the encryptor with additional means to control the decryption possible with the user secret key or the master secret key. Our token-based multi-input functional encryption (token-based MIFE) uses tokens both in the encryption and decryption processes, in addition to secret keys. Roughly speaking, in token-based MIFE, both an appropriate decryption token and the secret key are required to decrypt, and as the encryptors control the tokens, this provides an additional mechanism to address the above discussed issues. The purpose of this paper it to formalize token-based MIFE.

Token-Based MIFE. In this paper, we focus on multi-input functional encryption in the private key setting [GGG+14, BLR+14, ARW16, LL16, BKS16,

KLM+16, ABDP15, DOT18]. In token-based MIFE, we introduce new parameters – encryption tokens *etk* and decryption tokens *dtk* – which are output by a token generation algorithm `GenToken`. Informally speaking, token-based MIFE scheme is similar to ordinary MIFE scheme, except that (1) an encryption token *etk* is added to the input of encryption algorithm `Enc`, and (2) decryption algorithm `Dec` takes as input a decryption token in addition to the user secret key (and ciphertext). The encryption and decryption tokens are intended to be generated by the users encrypting the data in question, and the generation can be done independently of the key generation server holding the master secret key. Furthermore, new tokens can be generated as frequently as desired, which allow the user to partition the data they encrypt. By only distributing the relevant decryption tokens to the relevant decryption servers/analysts, the users can control what part of the data is accessible. For example, by encrypting data for two different analysts using different tokens, the user can ensure that one analyst cannot access data intended for the other.

In principle, users could generate a new set of tokens for each encryption done, which would correspond to a very fine-grained partitioning. However, the overhead of generating and managing tokens might make this undesirable. Furthermore, at the time of encryption, it might not be clear how the data should be partitioned. To address this, we additionally consider the ability to restrict decryption tokens to only work for a specified set of encryptions, even if all encryptions have been done using the same encryption token.

Specific Token-Based MIFE Schemes. In this paper, we present three specific token-based MIFE schemes. Firstly, we construct a simple scheme by combining a MIFE scheme and an ordinary public key encryption (PKE) scheme. In this scheme, the encryption and decryption tokens *etk* and *dtk* correspond to a public and private key of the PKE scheme, and encryptions simply correspond to double encryptions, using the MIFE scheme as the inner encryption. This scheme allows the entity generating the tokens, e.g. a chosen user, to broadcast the encryption token to the other users over a public channel.

The second token-based MIFE is constructed from a pseudorandom function (PRF) and a MIFE scheme. In this scheme, the ciphertexts of the underlying MIFE scheme are masked with masks generated using the PRF. However, to ensure security, the scheme is required to be stateful. In contrast to the first scheme, the token generation can be run in a distributed manner in the sense that each user can run the corresponding part of the token generation independently of the other users, but is then required to send the generated part of the decryption token to the decryption server.

Lastly, the third construction is an extension of the second one using a constrained PRF as opposed to an ordinary one. This allows the scheme to support decryption token restriction. Furthermore, using the GGM tree-based PRF [GGM84] as a constrained PRF, allows an efficient instantiation.

1.3 Paper Organization

The rest of the paper is organized as follows: In Sect. 2 we review the cryptographic preliminaries. In Sect. 3, we introduce the formal syntax and security definitions of token-based MIFE. In Sect. 4, we show our specific *stateless* scheme and it's security. In Sects. 5 and 6, we show our specific efficient *stateful* schemes and their security.

2 Preliminaries

In the following, we introduce the notion used in the paper, as well as the primitives our constructions are based on. In addition to the primitives in this section, we make use of a standard public key encryption scheme, which is defined in Appendix A.

2.1 Notation

Throughout the paper we will use $\lambda \in \mathbb{N}$ to denote the security parameter and will sometimes suppress the dependency on λ, when λ is clear from the context. We denote by $y \leftarrow x$ the assignment of y to x, and by $s \leftarrow S$ we denote the selection of an element s uniformly at random from the set S. The notation $[n]$ represents the set $\{1, 2, \ldots, n\}$, and for $n_1 < n_2$, $[n_1; n_2]$ represents the set $\{n_1, n_1 + 1, \ldots, n_2\}$. For an algorithm $\mathcal{A}$, we denote by $y \leftarrow \mathcal{A}(x)$ that A is run with input x, and that the output is assigned to y.

2.2 Pseudorandom Function

A pseudorandom function (PRF) $\mathtt{F} : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ with keyspace $\mathcal{K}$, domain $\mathcal{D}$, and range $\mathcal{R}$, is given by the following two algorithms.

$\mathtt{F.KeyGen}(1^\lambda)$ This is the key generation algorithm which, on input the security parameter 1^λ, returns a key $k \in \mathcal{K}$.

$\mathtt{F.Eval}(k, x)$ This is the evaluation algorithm which, given key $k \in \mathcal{K}$ and input $x \in D$, returns an output value $y \in R$.

Security is defined via the security game shown in Fig. 1.

Definition 1. *Let the advantage of an adversary $\mathcal{A}$ playing the security game in Fig. 1 with respect to a pseudorandom function $F = (\mathtt{KeyGen}, \mathtt{Eval})$ be defined as*

$$\mathtt{Adv}^{\mathtt{PRF}}_{F,\mathcal{A}}(\lambda) = 2\left|\Pr[\mathrm{PRF}^{F}_{\mathcal{A}}(\lambda) \Rightarrow 1] - \frac{1}{2}\right|.$$

F is said to be secure if for all PPT adversaries $\mathcal{A}$, $\mathtt{Adv}^{\mathtt{PRF}}_{F,\mathcal{A}}(\lambda)$ is negligible in the security parameter λ.

$\underline{\mathrm{PRF}^{\mathtt{F}}_{\mathcal{A}}(\lambda)}$:
$k \leftarrow \mathtt{F.KeyGen}(1^\lambda)$
$b \leftarrow_\$ \{0,1\}$
$\mathcal{F} \leftarrow \emptyset$
$b' \leftarrow \mathcal{A}^{\textsc{Eval}}(1^\lambda)$
return $(b = b')$

$\underline{\text{proc. } \textsc{Eval}(x)}$:
if $b = 1$
  $y \leftarrow \mathtt{F.Eval}(k, x)$
else
  if $\mathcal{F}[x] = \bot$, then $\mathcal{F}[x] \leftarrow_\$ R$
  $y \leftarrow \mathcal{F}[x]$
return y

Fig. 1. Game defining security of a pseudorandom function.

$\underline{\mathrm{PRF}^{\mathtt{F}}_{\mathcal{A}}(\lambda)}$:
$(S, st) \leftarrow \mathcal{A}(1^\lambda)$
$k \leftarrow \mathtt{F.KeyGen}(1^\lambda)$
$k_S \leftarrow \mathtt{F.Constrain}(k, S)$
$b \leftarrow_\$ \{0,1\}$
$\mathcal{F} \leftarrow \emptyset$
$b' \leftarrow \mathcal{A}^{\textsc{Eval}}(st, k_S)$
return $(b = b')$

$\underline{\text{proc. } \textsc{Eval}(x)}$:
if $x \in S$ return $\bot$
if $b = 1$
  $y \leftarrow \mathtt{F.Eval}(k, x)$
else
  if $\mathcal{F}[x] = \bot$, then $\mathcal{F}[x] \leftarrow_\$ R$
  $y \leftarrow \mathcal{F}[x]$
return y

Fig. 2. Game defining security of a constrained pseudorandom function.

Constrained Pseudorandom Function. A constrained PRF is an extension of an ordinary PRF that allows PRF keys to be constrained to only be usable for certain inputs. Specifically, besides the algorithms `KeyGen` and `Eval` defined for an ordinary PRF, a constrained PRF additionally includes the following algorithm:

$\mathtt{Constrain}(k, S)$ Given a key $k \in \mathcal{K}$ and a set $S \subset \mathcal{D}$, this constraining algorithm returns a constrained key k_S.

For correctness, it is required that for all security parameters λ, all keys $k \leftarrow \mathtt{KeyGen}(1^\lambda)$, all sets $S \subset \mathcal{D}$, all constrained keys $k_S \leftarrow \mathtt{Puncture}(k, S)$, it holds that

$$\mathtt{Eval}(k_S, x) = \begin{cases} \mathtt{Eval}(k, x) & \text{if } x \in S \\ \bot & \text{otherwise} \end{cases}$$

We define (selective) security for a constrained PRF $\mathtt{F}$ via the game shown in Fig. 2.

Definition 2. *Let the advantage of an adversary $\mathcal{A}$ playing the security game in Fig. 1 with respect to a pseudorandom function $\mathtt{F} = (\mathtt{KeyGen}, \mathtt{Eval})$ be defined as*

$$\mathtt{Adv}^{\mathtt{PRF}}_{F,\mathcal{A}}(\lambda) = 2\left|\Pr[\mathrm{PRF}^{F}_{\mathcal{A}}(\lambda) \Rightarrow 1] - \frac{1}{2}\right|.$$

F is said to be secure if for all PPT adversaries $\mathcal{A}$, $\mathtt{Adv}^{\mathtt{PRF}}_{F,\mathcal{A}}(\lambda)$ is negligible in the security parameter λ.

Note the the above defined security notion for a constrained PRF is slightly weaker than the notion considered by [BW13], as the adversary is only given access to a challenge evaluation oracle returning real or random values, as opposed to both an ordinary evaluation oracle alway returning $\mathtt{Eval}(k, x)$ and a challenge evaluation oracle which can be evaluated on distinct inputs. However, the above defined notion is sufficient to guarantee security for in our construction based on a constrained PRF.

2.3 Symmetric-Key Multi-input Functional Encryption

A symmetric-key multi-input functional encryption (MIFE) scheme $\mathtt{M}$ for a function f is given by the following algorithms [BLR+14].

$\mathtt{Setup}(1^\lambda)$ Given the security parameter λ, this setup algorithm returns public parameters mpk and a private master key msk.

$\mathtt{KeyGen}(msk, y)$ Given msk and value y, this key generation algorithm returns a secret key sk_y.

$\mathtt{Enc}(msk, x)$ Given the master key msk, an index i, and a message x_i, this encryption algorithm returns a ciphertext c_i.

$\mathtt{Dec}(sk_y, c_1, \ldots, c_n)$ Given sk_y and ciphertexts $(c_1, \ldots, c_n)$ encrypting message vectors $x_1, \ldots, x_n$, this decryption algorithm returns either $f(x_1, \ldots, x_n, y)$ or the error symbol $\perp$.

Correctness is defined in the obvious way. Adaptive security of a MIFE scheme $\mathtt{M}$ is defined via the following security game.

$\underline{\mathrm{IND}^\beta_{\mathtt{M},\mathcal{A}}(\lambda)}$:
$(mpk, msk) \leftarrow \mathtt{Setup}(1^\lambda)$
$b \leftarrow \mathcal{A}^{\textsc{KeyGen},\textsc{Enc}}(mpk)$
return b

$\underline{\text{proc. } \textsc{KeyGen}(y)}$:
$sk_y \leftarrow \mathtt{KeyGen}(msk, y)$
return sk_y

$\underline{\text{proc. } \textsc{Enc}(i, x_i^0, x_i^1)}$:
$c \leftarrow \mathtt{Enc}(msk, i, x_i^\beta)$
return c

In the above game it is required that for all $j_1, \ldots, j_n \in [Q_1] \times \cdots \times [Q_n]$, where for all $i \in [n]$, Q_i denotes the number of encryption queries for index i, $\mathcal{A}$ only makes queries y to KeyGen satisfying

$$f(x_1^{j_1,0}, \ldots, x_n^{j_n,0}, y) = f(x_1^{j_1,1}, \ldots, x_n^{j_n,1}, y)$$

where $(x_i^{j,0}, x_i^{j,1})$ denotes the values submitted by $\mathcal{A}$ in its jth query to Enc for index i.

Definition 3. *A scheme MIFE $\mathtt{M}$ is said to be $\mathtt{IND}$ secure, if for all PPT algorithms $\mathcal{A}$, the advantage*

$$\mathtt{Adv}^{\mathtt{IND}}_{M,\mathcal{A}}(\lambda) = \left|\Pr[IND^0_{M,\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[IND^1_{M,\mathcal{A}}(\lambda) \Rightarrow 1]\right|$$

is negligible in the security parameter λ.

3 Token-Based MIFE

3.1 Functionality

A token-based MIFE extends the functionality of an ordinary MIFE as defined in Sect. 2.3, by including an additional algorithm `GenToken` which, on input the parameters mpk of the scheme, generates encryption tokens $etk_1, \ldots, etk_n$ and a decryption token dtk. Each user i will then use encryption token etk_i as an additional input to the encryption algorithm, while the decryption server will use dtk as an additional input to the decryption algorithm. For simplicity, we will in our formalization consider a setup in which the evaluation of the functionality f implemented by the token-based MIFE is done over values $x_1, \ldots, x_n$ where x_i is encrypted by user i using encryption token etk_i. Lastly, our formalization considers token-based MIFE schemes supporting restricting decryption tokens to only work for specific ciphertexts. This is captured via the `Restrict` algorithm, which takes as input a decryption token dtk and a set S consisting of index pairs (i, j) referring to the jth encryption of the ith input, and outputs a decryption token dtk_S that only works for ciphertexts specified by S.

More formally, a token-based MIFE scheme for functionality $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \times \mathcal{Y} \to \mathcal{R}$ and a class $\mathcal{S} \subseteq 2^{[n]\times\mathbb{N}}$ of supported restriction sets, is given by the following algorithms.

`Setup` (1^λ) Given the security parameter λ, this setup algorithm returns a secret master key msk and public parameters mpk.

`KeyGen` (msk, y) Given msk and value y, this key generation algorithm returns a secret key sk_y.

`GenToken` (mpk) Given the public parameters mpk, this token generation algorithm returns encryption tokens $etk_1, \ldots, etk_u$ and a decryption token dtk.

`Enc` (msk, i, etk_i, x_i) Given the master key msk, a slot index i, a corresponding encryption token etk_i, and a message x_i, this encryption algorithm returns a ciphertext c_i and an updated encryption token etk'_i.

`Restrict` (dtk, S) Given dtk and a set S of index pairs, each pair $(i, j) \in S$ referring to the jth ciphertext encrypted for slot i, this algorithm returns a restricted decryption token dtk_S.

`Dec`$(dtk, sk_y, c_1, \ldots, c_n)$ Given dtk, sk_y, and ciphertexts $c_1, \ldots, c_n$ encrypting messages $x_1, \ldots, x_n$, this decryption algorithm returns either $f(x_1, \ldots, x_n, y)$ or the error symbol $\perp$.

Stateful/Stateless Schemes. Note that the above definition allows the encryption algorithm `Enc` to be stateful in the sense that, in addition to the ciphertext c, `Enc` returns an updated encryption token etk'_i. The premise is that user i will use the updated token etk'_i (as opposed to the old encryption token etk_i) in the following encryption. We refer to this type of scheme as a *stateful* scheme. However, we will additionally consider *stateless* schemes in which the encryption token is not updated, that is, for $(c_i, etk'_i) \leftarrow$ `Enc`(msk, i, etk_i, x_i) it holds that $etk'_i = etk_i$ regardless of the other inputs msk, i, and x_i.

Support for Decryption Token Restriction. While the above definition allows decryption tokens to be restricted via the `Restrict` algorithm, we will also consider scheme that essentially do not support this. In this case, we simply set the supported class of restriction sets $\mathcal{S}$ to be $\emptyset$ and let `Restrict` return an empty string. In particular note that for a stateless scheme, the input to the encryption algorithm besides x_i will remain the same, which implies that decryption tokens cannot be meaningfully restricted, assuming the ability to decrypt is independent of x_i. Hence, we will only consider decryption token restriction for stateful schemes.

Correctness for Stateless Schemes. For a token-based MIFE scheme $\mathtt{T}$ for function f, we require that for all security parameters λ, any input $x_1, \ldots, x_n \in \mathcal{X}$, any $y \in \mathcal{Y}$, any $(mpk, msk) \leftarrow \mathtt{T.Setup}(1^\lambda)$, any $sk_y \leftarrow \mathtt{T.KeyGen}(msk, y)$, any $(etk_1, \ldots, etk_u, dtk) \leftarrow \mathtt{T.GenToken}(mpk)$, any $i \in [n]$, and any $c_i \leftarrow \mathtt{T.Enc}(msk, i, etk_i, x_i)$, it holds that

$$\mathtt{T.Dec}(sk_y, dtk, c_1, \ldots, c_n) = f(x_1, \ldots, x_n, y).$$

Correctness for Stateful Schemes. For a token-based MIFE scheme $\mathtt{T}$ for function f, we require that for all security parameters λ, any set of n values $s_1, \ldots, s_n$ polynomial in λ, any set of $n(s_1 + \cdots + s_n)$ inputs, namely, $x_1^{(1)}, \ldots, x_1^{(s_1)} \in \mathcal{X}_1; \ldots; x_1^{(s_n)}, \ldots, x_n^{(s_n)} \in \mathcal{X}_n$, any $y \in \mathcal{Y}$, any $(mpk, msk) \leftarrow \mathtt{T.Setup}(1^\lambda)$, any $sk_y \leftarrow \mathtt{T.KeyGen}(msk, y)$, any set of encryption/decryption tokens $(etk_1, \ldots, etk_u, dtk) \leftarrow \mathtt{T.GenToken}(mpk)$, any set of ciphertexts $(c_1^{(j_1)}, \ldots, c_n^{(j_n)})$ obtained by, for each $i \in [n]$, iteratively computing $(c_i^{(j)}, etk_i^{(j+1)}) \leftarrow \mathtt{T.Enc}(msk, i, etk_i^{(j)}, x_i^{(j)})$, $j \in [s_i]$, where $etk_i^{(1)} = etk_i$, it holds that

$$\mathtt{T.Dec}(sk_y, dtk, c_1^{(s_1)}, \ldots, c_n^{(s_n)}) = f(x_1^{(s_1)}, \ldots, x_n^{(s_n)}, y)$$

and for all sets $S \in \mathcal{S}$ for which $(1, j_1), \ldots, (n, j_n) \in S$, and all $dtk_S \leftarrow \mathtt{T.Restrict}(dtk, S)$, it likewise holds that

$$\mathtt{T.Dec}(sk_y, dtk_S, c_1^{(s_1)}, \ldots, c_n^{(s_n)}) = f(x_1^{(s_1)}, \ldots, x_n^{(s_n)}, y)$$

Remark 1. Note that correctness for stateless schemes is in fact captured as a special case of correctness for stateful schemes. However, we explicitly included the former for readability, since it is simpler than the latter.

3.2 Security

FE-IND Security. Firstly, we consider security against a malicious decryption server who attempts to derive additional information regarding plaintexts x_i beyond what is revealed by $f(x_1, \ldots, x_n, y)$ obtained in an honest decryption. This is captured by the security notion functional encryption indistinguishability (FE-IND) defined via the following game for a scheme $\mathtt{T}$. In the game, the adversary will be given an unrestricted decryption token, a key generation oracles, as well as a (stateful) challenge encryption oracle. The FE-IND notion mirrors the IND notion defined for an ordinary MIFE (see Sect. 2.3).

$\text{FE-IND}^{\beta}_{\mathtt{T},\mathcal{A}}(\lambda)$:
$(mpk, msk) \leftarrow \mathtt{Setup}(1^\lambda)$
$(etk_1, \ldots, etk_n, dtk) \leftarrow \mathtt{GenToken}(mpk)$
$b \leftarrow \mathcal{A}^{\text{KeyGen},\text{Enc}}(mpk, dtk)$
return b

proc. KeyGen(y):
return $sk_y \leftarrow \mathtt{KeyGen}(msk, y)$

proc. Enc(i, x_i^0, x_i^1):
$(c, etk_i') \leftarrow \mathtt{Enc}(msk, etk_i, i, x_i^\beta)$
$etk_i \leftarrow etk_i'$
return c

In the above game it is required that for all $j_1, \ldots, j_n \in [Q_1] \times \cdots \times [Q_n]$, where for all $i \in [n]$, Q_i denotes the number of encryption queries for index i, $\mathcal{A}$ only makes queries y to KeyGen satisfying

$$f(x_1^{j,0}, \ldots, x_n^{j,0}, y) = f(x_1^{j,1}, \ldots, x_n^{j,1}, y)$$

where $(x_i^{j,0}, x_i^{j,1})$ denotes the values submitted by $\mathcal{A}$ in its jth query to Enc for index i.

Definition 4. *A scheme token-based MIFE* `T` *is said to be* `FE-IND` *secure, if for all PPT algorithms* $\mathcal{A}$*, the advantage*

$$\mathtt{Adv}^{\mathtt{FE-IND}}_{T,\mathcal{A}}(\lambda) = \left|\Pr[FE - IND^0_{T,\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[FE - IND^1_{T,\mathcal{A}}(\lambda) \Rightarrow 1]\right|$$

is negligible in the security parameter λ*.*

TK-IND Security. We additionally consider security against a malicious decryption server who attempts to learn any information regarding plaintexts x_i without possessing the appropriate decryption token. This is captured by the security notion token-based indistinguishability (TK-IND) defined via the following game for scheme `T`. Note that in the security game, the adversary $\mathcal{A}$ is given the master secret key msk as input, and hence captures a malicious decryption server colluding with the key generation server. Furthermore, $\mathcal{A}$ is given a restricted decryption token dtk_S for a set S of his own choice, but is required to submit challenge queries that are not covered by S. This captures that dtk_S does not leak information that would assist decryption of ciphertexts not covered by S. Note that our notion is selective in terms of the choice of S, as $\mathcal{A}$ is required to commit to S before interacting with the challenge encryption oracle. Lastly note that for a scheme not supporting decryption token restriction, the `Restrict` algorithm is implicitly defined to always return an empty string, which implies that $\mathcal{A}$ would only receive msk as input.

$\text{TK-IND}^{\beta}_{\mathtt{T},\mathcal{A}}(\lambda)$:
$(mpk, msk) \leftarrow \mathtt{Setup}(1^\lambda)$
$(\mathcal{S}, st) \leftarrow \mathcal{A}(mpk, msk)$
$(etk_1, \ldots, etk_l, dtk) \leftarrow \mathtt{GenToken}(mpk)$
$j_1, \ldots, j_n \leftarrow 0$
$dtk_{\mathcal{S}} \leftarrow \mathtt{Restrict}(dtk, \mathcal{S})$
$b \leftarrow \mathcal{A}^{\text{Enc}}(st, dtk_{\mathcal{S}})$
return b

proc. Enc(i, x_i^0, x_i^1):
if $(i, j_i) \in \mathcal{S}$
  return $\perp$
$(c, etk_i') \leftarrow \mathtt{Enc}(msk, etk_i, i, x_i^\beta)$
$j_i \leftarrow j_i + 1$
$etk_i \leftarrow etk_i'$
return c

Definition 5. *A token-based MIFE scheme* T *is said to be* TK-IND *secure, if for all PPT algorithms* $\mathcal{A}$*, the advantage*

$$\mathtt{Adv}^{\mathtt{TK-IND}}_{T,\mathcal{A}}(\lambda) = \left|\Pr[TK-IND^0_{T,\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[TK-IND^1_{T,\mathcal{A}}(\lambda) \Rightarrow 1]\right|$$

is negligible in the security parameter λ*.*

Besides the above, we consider a slightly stronger security notion which captures schemes in which the encryption tokens are broadcast to the users over a public channel. We will refer to this notion as public token-based indistinguishability (pTK-IND). More specifically, we denote by pTK-IND a security game identical to the TK-IND game defined above, except that in the second invocation of the adversary $\mathcal{A}$, $etk_1, \ldots, etk_n$ will be given as input to $\mathcal{A}$ in addition to st and $dtk_{\mathcal{S}}$. Based on this game, pTK-IND security is defined as follows.

Definition 6. *A token-based MIFE scheme* T *is said to be* pTK-IND *secure, if for all PPT algorithms* $\mathcal{A}$*, the advantage*

$$\mathtt{Adv}^{\mathtt{pTK-IND}}_{T,\mathcal{A}}(\lambda) = \left|\Pr[pTK-IND^0_{T,\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[pTK-IND^1_{T,\mathcal{A}}(\lambda) \Rightarrow 1]\right|$$

is negligible in the security parameter λ*.*

4 A Stateless Scheme

We will now present a simple stateless token-based MIFE scheme based on a standard MIFE scheme and a public key encryption scheme. The construction uses a single public key for the encryption tokens, and simply constructs a double encryption of messages, using the public key encryption as the outer layer. This leads to a scheme in which e.g. a chosen user can generate the encryption/decryption tokens on behalf of all users, and then simply broadcast the encryption token to the remaining users (as well as provide the decryption token to the decryption server fi/when appropriate). However, as this is a stateless scheme, it will not support decryption token restriction.

Concretely, we construct a token-based MIFE scheme $\mathtt{T}$ for function f using an ordinary public-key encryption scheme $\mathtt{PKE} = (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ and a multi-input functional encryption scheme $\mathtt{M} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ for f as follows.

$\mathtt{Setup}\ (1^\lambda)$ Return $(mpk, msk) \leftarrow \mathtt{M.Setup}(1^\lambda)$.
$\mathtt{KeyGen}\ (msk, y)$ Return $sk_y \leftarrow \mathtt{M.KeyGen}(msk, y)$.
$\mathtt{GenToken}\ (\lambda)$ Compute $(pk, sk) \leftarrow \mathtt{PKE.KeyGen}(1^\lambda)$, set $etk_1 = \cdots = etk_l \leftarrow pk$ and $dtk \leftarrow sk$. Finally return $(etk_1, \ldots, etk_l, dtk)$.
$\mathtt{Enc}\ (msk, i, etk_i, x_i)$ Compute $c' \leftarrow \mathtt{M.Enc}(msk, i, x_i)$; $c \leftarrow \mathtt{PKE.Enc}(etk_i, c')$. Return (c, u).
$\mathtt{Dec}\ (dtk, sk_y, c_1, \ldots, c_n)$ For each $i \in [n]$, compute $c'_i \leftarrow \mathtt{PKE.Dec}(dtk, c_i)$. Return $\mathtt{M.Dec}(sk_y, c'_1, \ldots, c'_n)$.

4.1 Security

The security of the above construction is established via the following two theorems. The corresponding proofs can be found in Appendix B.

Theorem 1. *Assume the MIFE scheme* $\mathtt{M}$ *is IND secure. Then the above scheme* $\mathtt{T}$ *is FE-IND secure. Specifically, for every PPT adversary* $\mathcal{A}$ *against the FE-IND security of* $\mathtt{T}$*, there exists a PPT adversary* $\mathcal{B}$ *against the IND security of* $\mathtt{M}$ *such that* $\mathtt{Adv}^{\mathtt{FE-IND}}_{T,\mathcal{A}} \leq \mathtt{Adv}^{\mathtt{IND}}_{M,\mathcal{B}}$.

Theorem 2. *Assume* $\mathtt{PKE}$ *is IND-CPA secure. Then the above scheme* $\mathtt{T}$ *is pTK-IND secure. Specifically, for every PPT adversary* $\mathcal{A}$ *against the pTK-IND security of* $\mathtt{T}$*, there exists a PPT adversary* $\mathcal{B}$ *against the IND-CPA security of* $\mathtt{PKE}$ *such that* $\mathtt{Adv}^{\mathtt{TK-IND}}_{T,\mathcal{A}} \leq n \cdot \mathtt{Adv}^{\mathtt{IND-CPA}}_{PKE,\mathcal{B}}$.

5 A Stateful Scheme

We will now present a stateful token-based MIFE scheme based on a standard MIFE scheme and a PRF. Compared to the stateless scheme in the previous section, the computational overhead of making the MIFE scheme token-based is much lower, as a PRF can be implemented much more efficiently compared to a PKE. Furthermore, the scheme allows the `GenToken` algorithm to be computed in a distributed manner; each user will be able to independently generate his own encryption token etk_i and the corresponding part of the decryption token dtk_i. For the decryption server to be able to decrypt, it is then required that each user sends dtk_i to the server, which will then form the full decryption token $dtk = (dtk_1, \ldots, dtk_n)$. Like the stateless scheme, the construction idea is simple; the ciphertexts of the MIFE are simply masked with a mask generated using the PRF evaluated. To guarantee security, the scheme must be stateful, as it must be ensured that the same mask is never used twice.

Concretely, we construct a token-based MIFE scheme $\mathtt{T}$ using a PRF $\mathtt{PRF} = \{\mathtt{KeyGen}, \mathtt{Eval}\}$ with range $\{0,1\}^l$, and a MIFE scheme $\mathtt{M} = \{\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec}\}$ with ciphertext space $\mathcal{C} \subset \{0,1\}^l$ as follows.

$\mathtt{Setup}$ (1^λ) Return $(mpk, msk) \leftarrow \mathtt{M.Setup}(1^\lambda)$.

$\mathtt{KeyGen}$ (msk, y) Return $sk_y \leftarrow \mathtt{M.KeyGen}(msk, y)$.

$\mathtt{GenToken}$ (λ) For $i \in [n]$ compute $k_i \leftarrow \mathtt{PRF.KeyGen}(1^\lambda)$, and set $etk_i \leftarrow (k_i, 0)$. Finally set $dtk \leftarrow (k_1, \ldots, k_n)$, and return $(etk_1, \ldots, etk_n, dtk)$.

$\mathtt{Enc}$ (msk, i, etk, x_i) Parse $etk \rightarrow (k, j)$, and compute $m \leftarrow \mathtt{PRF.Eval}(k, j)$. Then compute $c' \leftarrow \mathtt{M.Enc}(msk, i, x_i)$, and set $c \leftarrow (c' \oplus m, j)$, $j' \leftarrow j + 1$, and $etk' \leftarrow (k, u, j')$. Finally return (c, etk').

$\mathtt{Dec}$ $(dtk, sk_y, c_1, \ldots, c_n)$ Parse $dtk \rightarrow (k_1, \ldots, k_l)$ and $c_i \rightarrow (c'_i, j_i)$ for $i \in [n]$. For each $i \in [n]$, compute $m_i \leftarrow \mathtt{PRF.Eval}(k_i, j_i)$ and $c''_i \leftarrow c'_i \oplus m_i$. Finally return $z \leftarrow \mathtt{M.Dec}(sk'_y, c''_1, \ldots, c''_n)$.

5.1 Security

The security of the above construction is established via the following two theorems.

Theorem 3. *Assume the MIFE scheme* $\mathtt{M}$ *is IND secure. Then the above scheme* $\mathtt{T}$ *is FE-IND secure. Specifically, for every PPT adversary* $\mathcal{A}$ *against the FE-IND security of* $\mathtt{T}$*, there exists a PPT adversary* $\mathcal{B}$ *against the IND security of* $\mathtt{M}$ *such that* $\mathtt{Adv}^{\mathtt{TK-IND}}_{T,\mathcal{A}} \leq \mathtt{Adv}^{\mathtt{IND}}_{M,\mathcal{B}}$.

Proof. The proof is a simple and straightforward reduction. Given adversary $\mathcal{A}$ against the FE-IND security of $\mathtt{T}$, we construct adversary $\mathcal{B}$ against the IND security of $\mathtt{M}$ as follows.

Initially, $\mathcal{B}$ is given parameters mpk which $\mathcal{B}$ simply forwards to $\mathcal{A}$. Furthermore, $\mathcal{B}$ will compute PRF keys $k_i \leftarrow \mathtt{P.KeyGen}(1^\lambda)$ and set variables $j_i \leftarrow 0$ for $i \in [n]$. When $\mathcal{A}$ submits a key generation query y, $\mathcal{B}$ simply forwards y to his own KeyGen oracle, and returns the response sk_y to $\mathcal{A}$. When $\mathcal{A}$ makes an encryption query (u, i, x_i^0, x_i^1), $\mathcal{B}$ forwards (i, x_i^0, x_i^1) to his own Enc oracle to obtain ciphertext c'. Then $\mathcal{B}$ computes $m \leftarrow \mathtt{P.Eval}(k_i, j_i)$, sets $c \leftarrow (c' \oplus m, j_i)$ and $j_i \leftarrow j_i + 1$, and lastly returns c to $\mathcal{A}$. Eventually $\mathcal{A}$ will terminate with output b, which $\mathcal{B}$ forwards as his own output.

By inspection, it should be clear that $\mathcal{B}$ provides a perfect simulation of the FE-IND game for $\mathcal{A}$, and that $\mathcal{B}$ wins the IND game for $\mathtt{M}$ (i.e. correctly guesses the challenge bit β) whenever $\mathcal{A}$ wins the FE-IND game for $\mathtt{T}$. Hence the theorem follows.

Theorem 4. *Assume the PRF* $\mathtt{P}$ *is secure. Then the above scheme* $\mathtt{T}$ *is TK-IND secure. Specifically, for every PPT adversary* $\mathcal{A}$ *against the TK-IND security of* $\mathtt{T}$*, there exists a PPT adversaries* $\mathcal{B}_1, \ldots, \mathcal{B}_l$ *against the PRF security of* $\mathtt{P}$ *such that* $\mathtt{Adv}^{\mathtt{TK-IND}}_{T,\mathcal{A}} \leq \mathtt{Adv}^{\mathtt{PRF}}_{P,\mathcal{B}_1} + \ldots + \mathtt{Adv}^{\mathtt{PRF}}_{P,\mathcal{B}_{2l}}$.

Proof. The proof is a series of simple game hops, firstly replacing the output of the PRF $\mathtt{P}$ with random values for each user, then changing the challenge bit β used by the encryption oracle, and finally replacing the output of $\mathtt{P}$ back to the real values. Note that since the scheme does not support decryption token restrictions, the restricted decryption token dtk_S will correspond to $\perp$ regardless of the set S, and we can ignore this input to $\mathcal{A}$ (as well as S output by $\mathcal{A}$).

More concretely, let G_0 denote the $TK - IND^0_{\mathtt{T},\mathcal{A}}$ game, and let G_t, $t \in [n]$, denote modifications of this game in which the encryption done in response to encryption queries (i, x_i^0, x_i^1), is modified as follows: if $i \leq t$, set $m \leftarrow \{0,1\}^t$, otherwise set $m \leftarrow \mathtt{P.Eval}(k_i, j_i)$. Furthermore, let G_{n+1} denote a modification of G_n in which the encryption oracle uses $\beta = 1$. Finally, let G_t for $t \in [n+2; 2n+1]$ denote modifications of G_{n+1} corresponding to reversing the changes introduced in games $G_1, \ldots, G_n$ i.e. the random masks m used in the response to encryption queries are replaced with $m \leftarrow \mathtt{P.Eval}(k_i, j_i)$ for each user i in turn, starting from user 1. It should be clear the game G_{2n+1} is identical to $\mathtt{TK} - \mathtt{IND}^1_{\mathtt{T},\mathcal{A}}$.

Via a simple reduction, we bound the difference between the probability that $\mathcal{A}$ outputs 1 in game G_t and in game G_{t+1}, $t \in \{0, \ldots, n-1\}$, with $\mathtt{Adv}^{\mathtt{PRF}}_{\mathtt{P},\mathcal{B}_t}$ for a

PPT algorithm $\mathcal{B}_t$. Specifically, $\mathcal{B}_t$ is constructed as follows. Initially, $\mathcal{B}_t$ generates $(mpk, msk) \leftarrow \mathtt{M.Setup}(1^\lambda)$, sets $j_1, \ldots, j_n \leftarrow 0$, computes $k_i \leftarrow \mathtt{P.KeyGen}(1^\lambda)$ for all $i \neq t$ and $i \in [n]$, and forwards (mpk, msk) to $\mathcal{A}$ (note that (k_i, j_i) corresponds to the encryption token etk_i for user i). For encryption queries (i, x_1^0, x_i^1) where $i \leq t$, $\mathcal{B}_t$ responds by using a randomly chosen mask $m_i \leftarrow \{0,1\}^l$. For queries where $i > t+1$, $\mathcal{B}_t$ responds using $m_i \leftarrow \mathtt{P.Eval}(k_u, i||j)$ and updating $j_i \leftarrow j_i+1$. However, for queries where $i = t+1$, $\mathcal{B}_t$ forwards j_i to his own $\mathtt{Eval}$ oracle to obtain m_i, and sets $j_i \leftarrow j_i + 1$. Finally, when $\mathcal{A}$ returns a bit b', $\mathcal{B}_t$ forwards this as his own output.

From the above description, it should be clear that $\mathcal{B}_t$ provides a perfect simulation of game G_t for $\mathcal{A}$ if the challenge bit β in the PRF security game played by $\mathcal{B}_t$ is 0. On the other hand, if $\beta = 1$, $\mathcal{B}_t$ provides a perfect simulation of game G_{t+1}. Hence, it directly follows that $|\Pr[G_t \Rightarrow 1] - \Pr[G_{t+1} \Rightarrow]| \leq \mathtt{Adv}^{\mathtt{PRF}}_{\mathtt{P},\mathcal{B}_t}$, $t \in [0; l-1]$. Furthermore, since in game G_n and G_{n+1}, the masks used in the response to encryption queries are picked uniformly at random, the distributions of $(m_i \oplus c_i', j_i)$ in the two games are identical, even though c_i' encrypts x_i^0 in game G_n and x_i^1 in game G_{n+1}. Finally, using an identical argument to the above, it follows that $|\Pr[G_i \Rightarrow 1] - \Pr[G_{i+1} \Rightarrow]| \leq \mathtt{Adv}^{\mathtt{PRF}}_{\mathtt{P},\mathcal{B}_i}$, $i \in [l+1; 2l+1]$.

Combining the above bounds, we obtain

$$\begin{aligned}
\mathtt{Adv}^{\mathtt{TK-IND}}_{\mathtt{T},\mathcal{A}} &= |\Pr[TK - IND^0_{\mathtt{T},\mathcal{A}} \Rightarrow 1] - TK - IND^1_{\mathtt{T},\mathcal{A}} \Rightarrow 1]| \\
&\leq \sum_{t=0}^{2l} |\Pr[G_t \Rightarrow 1] - \Pr[G_{t+1} \Rightarrow 1]| \\
&\leq \mathtt{Adv}^{\mathtt{PRF}}_{\mathtt{P},\mathcal{B}_0} + \ldots + \mathtt{Adv}^{\mathtt{PRF}}_{\mathtt{P},\mathcal{B}_{l-1}} + \mathtt{Adv}^{\mathtt{PRF}}_{\mathtt{P},\mathcal{B}_{l+1}} + \ldots + \mathtt{Adv}^{\mathtt{PRF}}_{\mathtt{P},\mathcal{B}_{2l}}
\end{aligned}$$

□

6 A Stateful Scheme Supporting Decryption Tokens Restriction

We will now present an extension of the scheme from Sect. 5 that allows restricting decryption tokens. First observe that the stateful scheme from Sect. 5 actually supports a simple form of decryption token restriction. Specifically, note that the decryption server is required to recover the masks m_i to be able to decrypt, which is possible as the decryption token dtk contains the PRF keys k_i used to generate these. To restrict dtk to only be usable for a set of ciphertext specified by a given set S, e.g. $S = \{(1, j_1), \ldots, (n, j_n)\}$, it is possible to simply use the relevant masks as a restricted decryption token i.e. $dtk_S = \{m_i = \mathtt{Eval}(k_i, j_i)\}_{i \in [n]}$. It is not difficult to see that security for ciphertexts not described by S will be preserved. However, the disadvantage of this solution is that the size of the decryption token will equal the size of S.

To obtain a more efficient solution, we will make use of a constrained PRF to limit the ability of the decryption server to only be able to generate the masks required to decrypt the ciphertexts described by S. By choosing an appropriate

instantiation of the constrained PRF, the size of the decryption token can be reduced. We discuss the details of the instantiation below.

The scheme $\mathtt{rT}$ is based on a MIFE $\mathtt{M} = \{\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec}\}$ and a constrained PRF $\mathtt{cP} = \{\mathtt{KeyGen}, \mathtt{Eval}, \mathtt{Restrict}\}$. The algorithms $\mathtt{rT.Setup}$, $\mathtt{rT.KeyGen}$, $\mathtt{rT.GenToken}$, $\mathtt{rT.Enc}$, and $\mathtt{rT.Dec}$ are identical to the stateless scheme $\mathtt{T}$ in Sect. 4, but for clarity, all algorithms are described below.

$\mathtt{Setup}$ (1^λ) Return $(mpk, msk) \leftarrow \mathtt{M.Setup}(1^\lambda)$.
$\mathtt{KeyGen}$ (msk, y) Return $sk_y \leftarrow \mathtt{M.KeyGen}(msk, y)$.
$\mathtt{GenToken}$ (λ) For $i \in [n]$ compute $k_i \leftarrow \mathtt{cP.KeyGen}(1^\lambda)$, and set $etk_i \leftarrow (k_i, 0)$. Finally set $dtk \leftarrow (k_1, \ldots, k_n)$, and return $(etk_1, \ldots, etk_n, dtk)$.
$\mathtt{Enc}$ (msk, i, etk, x_i) Parse $etk \rightarrow (k, j)$, and compute $m \leftarrow \mathtt{cP.Eval}(k, j)$. Then compute $c' \leftarrow \mathtt{M.Enc}(msk, i, x_i)$, and set $c \leftarrow (c' \oplus m, j)$, $j' \leftarrow j + 1$, and $etk' \leftarrow (k, u, j')$. Finally return (c, etk').
$\mathtt{Restrict}$ (dtk, S) Parse $dtk \rightarrow (k_1, \ldots, k_n)$ and let $S_i = \{j | (i, j) \in S\}$. Set $k'_i \leftarrow \mathtt{cP.Restrict}(k_i, S_i)$ for $i \in [n]$, and return $dtk_S \leftarrow (k'_1, \ldots, k'_n)$.
$\mathtt{Dec}$ $(dtk, sk_y, c_1, \ldots, c_n)$ Parse $dtk \rightarrow (k_1, \ldots, k_l)$ and $c_i \rightarrow (c'_i, j_i)$ for $i \in [n]$. For each $i \in [n]$, compute $m_i \leftarrow \mathtt{cP.Eval}(k_i, j_i)$ and $c''_i \leftarrow c'_i \oplus m_i$. Finally return $z \leftarrow \mathtt{M.Dec}(sk'_y, c''_1, \ldots, c''_n)$.

It is relatively straightforward to confirm that the scheme is correct.

6.1 Security

Theorem 5. *Assume the MIFE scheme $\mathtt{M}$ is IND secure. Then the above scheme $\mathtt{rT}$ is FE-IND secure. Specifically, for every PPT adversary $\mathcal{A}$ against the FE-IND security of $\mathtt{rT}$, there exists a PPT adversary $\mathcal{B}$ against the IND security of $\mathtt{M}$ such that $\mathtt{Adv}^{\mathtt{TK-IND}}_{\mathtt{rT},\mathcal{A}} \leq \mathtt{Adv}^{\mathtt{IND}}_{\mathtt{M},\mathcal{B}}$.*

The proof of the above theorem is identical to the proof of Theorem 3

Theorem 6. *Assume the PRF $\mathtt{P}$ is secure. Then the above scheme $\mathtt{T}$ is TK-IND secure. Specifically, for every PPT adversary $\mathcal{A}$ against the TK-IND security of $\mathtt{T}$, there exists a PPT adversaries $\mathcal{B}_1, \ldots, \mathcal{B}_l$ against the PRF security of $\mathtt{P}$ such that $\mathtt{Adv}^{\mathtt{TK-IND}}_{\mathtt{T},\mathcal{A}} \leq \mathtt{Adv}^{\mathtt{cPRF}}_{\mathtt{P},\mathcal{B}_1} + \ldots + \mathtt{Adv}^{\mathtt{cPRF}}_{\mathtt{P},\mathcal{B}_{2l}}$.*

(*Proof Sketch*). The proof is almost identical to the proof of Theorem 4, so we will just highlight the differences. In fact, the only difference is that the adversary $\mathcal{A}$ will have to be given a correctly formed restricted decryption token dtk_S corresponding to the set S output by $\mathcal{A}$. Following the description of the scheme, this implies that in all games G_t, $i \in [2l + 1]$, $\mathcal{A}$ will be given $dtk_S = (k_{S_1}, \ldots, k_{S_n})$, where $S_i = \{j | (i, j) \in S\}$ and $k_{S_i} \leftarrow \mathtt{cP.Restrict}(dtk, S_i)$. Note that the adversary $\mathcal{B}_t$ constructed to bound the difference between games G_t and G_{t+1}, for $t \in 0, \ldots, l - 1$, will have access to all keys k_i for $i \neq t + 1$ $i \in [n]$, and can hence directly compute $k_{S_i} \leftarrow \mathtt{P.Constrain}(k_i, S_i)$. Furthermore, $\mathcal{B}_t$ will initially be given S and can derive $S_{t+1} = \{j | (t+1, j) \in S\}$, and since $\mathcal{B}_t$ will be interacting in the security game of the constrained PRF cP, $\mathcal{B}_t$ will be able to submit this

as the initial step, and obtain $k_{S_{t+1}} \leftarrow \texttt{cP.Restrict}(k_{t+1}, S_{t+1})$ for the challenge key k_{t+1}. The remaining part of the simulation is exactly as in the proof of Theorem 4, and a an identical bound on the advantage of $\mathcal{A}$ is obtained. □

6.2 Efficient Instantiation of a Constrained PRF

As also observed in several other works [BW13,BGI14,KPTZ13], a selective secure constrained PRF can be obtained directly from the GGM tree-based construction of a PRF [GGM84]. More precisely, the secret key k_s at an internal node associated with the string s in the tree, allows the PRF to be evaluated on strings with the prefix s i.e. k_s is a constrained key for the set of strings with prefix s. Using this construction in combination with the above token-based MIFE, leads to restricted decryption tokens dtk_S consisting of PRF keys k_s such that all values j in the set S is captured by a prefix s, but no value $j' \in S$ is. This construction is particularly efficient when the values j in S are consecutive values.

A Public Key Encryption

A public key encryption (PKE) scheme PKE is defined by three algorithms with the following functionality:

$\texttt{PKE.KeyGen}(1^\lambda)$ This is the key generations algorithm, which on input the security parameter 1^λ, returns a public/private key pair (pk, sk).

$\texttt{PKE.Enc}(par, pk, m)$ This is the encryption algorithm, which on input a public key pk and a message m, returns an encryption c of m under pk.

$\texttt{PKE.Dec}(par, sk, c)$ This is the decryption algorithm, which on input a private key sk and a ciphertext c, returns either a message m or the error symbol $\perp$.

We require that a PKE scheme satisfies *perfect correctness*, that is, for all λ, all $(pk, sk) \leftarrow \texttt{PKE.KeyGen}(1^\lambda)$, and all m, it holds that $\texttt{PKE.Dec}(sk, \texttt{PKE.Enc}(pk, m)) = m$.

$\underline{\text{IND-CPA}^{\text{PKE}}_{\mathcal{A}}(\lambda)}$:
$(pk^*, sk^*) \leftarrow \texttt{PKE.KeyGen}(1^\lambda)$
$b \leftarrow_\$ \{0,1\}$;
$(m_0, m_1, st) \leftarrow \mathcal{A}^{(}pk^*)$
$c^* \leftarrow \texttt{PKE.Enc}(pk^*, m_b)$
$b' \leftarrow \mathcal{A}(st, c^*)$
return $(b = b')$

Fig. 3. Game defining indistinguishability under chosen plaintext attacks (IND-CPA) for a PKE scheme.

The standard IND-CPA security notion for PKE scheme is defined via the game shown in Fig. 3.

Definition 7. *Let the advantage of an adversary $\mathcal{A}$ playing the IND-CPA game with respect to a PKE scheme PKE, be defined as:*

$$\mathtt{Adv}^{\mathtt{IND-CPA}}_{PKE,\mathcal{A}}(\lambda) = 2\left|\Pr[\mathit{IND\text{-}CPA}^{PKE}_{\mathcal{A}}(\lambda) \Rightarrow 1] - \frac{1}{2}\right|.$$

A scheme PKE is said to be IND-CPA secure, if for all PPT adversaries $\mathcal{A}$, $\mathtt{Adv}^{\mathtt{IND-CPA}}_{PKE,\mathcal{A}}(\lambda)$ is negligible in the security parameter λ.

B Security Proofs for Stateless Scheme

Theorem 7. *Assume the MIFE scheme M is IND secure. Then the above scheme T is FE-IND secure. Specifically, for every PPT adversary $\mathcal{A}$ against the FE-IND security of T, there exists a PPT adversary $\mathcal{B}$ against the IND security of M such that $\mathtt{Adv}^{\mathtt{FE-IND}}_{T,\mathcal{A}} \leq \mathtt{Adv}^{\mathtt{IND}}_{M,\mathcal{B}}$.*

Proof. The proof is a simple and straightforward reduction. Given adversary $\mathcal{A}$ against the FE-IND security of T, we construct adversary $\mathcal{B}$ against the IND security of M as follows.

Initially, $\mathcal{B}$ is given parameters mpk which $\mathcal{B}$ simply forwards to $\mathcal{A}$. Furthermore, $\mathcal{B}$ will compute $(pk, sk) \leftarrow \mathtt{PKE.KeyGen}(1^\lambda)$. When $\mathcal{A}$ submits a key generation query y, $\mathcal{B}$ simply forwards y to his own KeyGen oracle, and returns the response sk_y to $\mathcal{A}$. When $\mathcal{A}$ makes an encryption query (i, x_i^0, x_i^1), $\mathcal{B}$ forwards (i, x_i^0, x_i^1) to his own Enc oracle to obtain ciphertext c'. Then $\mathcal{B}$ computes $c \leftarrow \mathtt{PKE.Enc}(pk, c')$, and returns c to $\mathcal{A}$. Eventually $\mathcal{A}$ will terminate with output b', which $\mathcal{B}$ forwards as his own output.

By inspection, it should be clear that $\mathcal{B}$ provides a perfect simulation of the FE-IND game for $\mathcal{A}$, and that $\mathcal{B}$ wins the IND game for M (i.e. correctly guesses the challenge bit b) whenever $\mathcal{A}$ wins the FE-IND game for T. Hence the theorem follows.

Theorem 8. *Assume PKE is IND-CPA secure. Then the above scheme T is pTK-IND secure. Specifically, for every PPT adversary $\mathcal{A}$ against the pTK-IND security of T, there exists a PPT adversary $\mathcal{B}$ against the IND-CPA security of PKE such that $\mathtt{Adv}^{\mathtt{TK-IND}}_{T,\mathcal{A}} \leq n \cdot \mathtt{Adv}^{\mathtt{IND-CPA}}_{PKE,\mathcal{B}}$.*

Proof. Again, the proof is a simple and straightforward reduction. In the following, we will for convenience make use of the standard extension of IND-CPA security to the multi-challenge setting.

Given adversary $\mathcal{A}$ against the TK-IND security of T, we construct adversary $\mathcal{B}$ against the IND-CPA security of PKE as follows.

Initially, $\mathcal{B}$ is given parameters pk. $\mathcal{B}$ computes $(mpk, msk) \leftarrow \mathtt{M.Setup}(1^\lambda)$. and forwards (mpk, msk) to $\mathcal{A}$. When $\mathcal{A}$ submits a key generation query y, $\mathcal{B}$ simply computes $sk_y \leftarrow \mathtt{M.KeyGen}(msk, y)$ and returns sk_y to $\mathcal{A}$. When $\mathcal{A}$ makes an encryption query (i, x_i^0, x_i^1), $\mathcal{B}$ computes $c_i'^{(0)} \leftarrow \mathtt{M.Enc}(msk, i, x_i^0)$ and $c_i'^{(1)} \leftarrow \mathtt{M.Enc}(msk, i, x_i^1)$. $\mathcal{B}$ then submits $(c_1'^{(0)}, c_1'^{(1)}), \ldots, (c_n'^{(0)}, c_n'^{(1)})$ to

the multi-challenge IND-CPA challenge oracle to obtain the challenge vector $(c_1, \ldots, c_n)$. $\mathcal{B}$ then returns it to $\mathcal{A}$. Eventually $\mathcal{A}$ will terminate with output b', which $\mathcal{B}$ forwards as his own output.

By inspection, it should be clear that $\mathcal{B}$ provides a perfect simulation of the TK-IND game for $\mathcal{A}$, and that $\mathcal{B}$ wins the multi-challange IND-CPA game for `M` (i.e. correctly guesses the challenge bit b) whenever $\mathcal{A}$ wins the TK-IND game for `T`. Furthermore, since the multi-challenge IND-CPA security reduces to the normal IND-CPA security with reduction n, the number of challenges, the theorem follows. □

References

[ABDP15] Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. Cryptology ePrint Archive, Report 2015/017 (2015). http://eprint.iacr.org/2015/017

[ARW16] Abdalla, M., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. Cryptology ePrint Archive, Report 2016/425 (2016). http://eprint.iacr.org/2016/425

[BGI14] Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_29

[BKS16] Brakerski, Z., Komargodski, I., Segev, G.: Multi-input functional encryption in the private-key setting: stronger security from weaker assumptions. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 852–880. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_30

[BLR+14] Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., Zimmerman, J.: Semantically secure order-revealing encryption: multi-input functional encryption without obfuscation. Cryptology ePrint Archive, Report 2014/834 (2014). http://eprint.iacr.org/2014/834

[BW13] Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_15

[DOT18] Datta, P., Okamoto, T., Tomida, J.: Full-hiding (Unbounded) multi-input inner product functional encryption from the k-linear assumption. In: Abdalla, M., Dahab, R. (eds.) PKC 2018. LNCS, vol. 10770, pp. 245–277. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76581-5_9

[GGG+14] Goldwasser, S., et al.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_32

[GGM84] Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: 25th FOCS, pp. 464–479. IEEE Computer Society Press, October 1984

[KLM+16] Kim, S., Lewi, K., Mandal, A., Montgomery, H., Roy, A., Wu, D.J.: Function-hiding inner product encryption is practical. Cryptology ePrint Archive, Report 2016/440 (2016). http://eprint.iacr.org/2016/440

[KPTZ13] Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: Sadeghi, A.-R., Gligor, V.D., Yung, M. (eds.) ACM CCS 13, pp. 669–684. ACM Press, November 2013

[LL16] Lee, K., Lee, D.H.: Two-input functional encryption for inner products from bilinear maps. Cryptology ePrint Archive, Report 2016/432 (2016). http://eprint.iacr.org/2016/432