PhD-FSTC-2017-03
The Faculty of Sciences, Technology and Communication

DISSERTATION

Defense held on 17/01/2017 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Afonso DELERUE ARRIAGA

Born on 7 March 1984 in Angra do Heroísmo, Portugal

Private Functional Encryption – Hiding What Cannot Be Learned Through Function Evaluation

Dissertation defense committee

Abstract

Functional encryption (FE) is a generalization of many commonly employed cryptographic primitives, such as keyword search encryption (KS), identity-based encryption (IBE), inner-product encryption (IPE) and attribute-based encryption (ABE). In an FE scheme, the holder of a master secret key can issue tokens associated with functions of its choice. Possessing a token for f allows one to recover $f(m)$, given an encryption of m. As it is important that ciphertexts preserve data privacy, in various scenarios it is also important that tokens do not expose their associated function. A notable example being the usage of FE to search over encrypted data without revealing the search query. Function privacy is an emerging new notion that aims to address this problem. The difficulty of formalizing it lies in the verification functionality, as the holder of a token for function f may encrypt arbitrary messages using the public key, and obtain a large number of evaluations of f.

Prior privacy models in the literature were fine-tuned for specific functionalities, did not model correlations between ciphertexts and decryption tokens, or fell under strong uninstantiability results. Our first contribution is a new indistinguishability-based privacy notion that overcomes these limitations and is flexible enough to capture all previously proposed indistinguishability-based definitions as particular cases.

The second contribution of this thesis is five constructions of private functional encryption supporting different classes of functions and meeting varying degrees of security: (1) a white-box construction of an Anonymous IBE scheme based on composite-order groups, shown to be secure in the absence of correlated messages; (2) a simple and functionality-agnostic black-box construction from obfuscation, also shown to be secure in the absence of correlated messages; (3) a more evolved and still functionality-agnostic construction that achieves a form of function privacy that tolerates limited correlations between messages and functions; (4) a KS scheme achieving privacy in the presence of correlated messages beyond all previously proposed indistinguishability-based security definitions; (5) a KS construction that achieves our strongest notion of privacy (but relies on a more expressive form of obfuscation than the previous construction).

The standard approach in FE is to model complex functions as circuits, which yields inefficient evaluations over large inputs. As our third contribution, we propose a new primitive that we call “updatable functional encryption” (UFE), where instead of circuits we deal with RAM programs, which are closer to how programs are expressed in von Neumann architecture. We impose strict efficiency constrains and we envision tokens that are capable of updating the ciphertext, over which other tokens can be subsequently executed. We define a security notion for our primitive and propose a candidate construction from obfuscation, which serves as a starting point towards the realization of other schemes and contributes to the study on how to compute RAM programs over public-key encrypted data.

Acknowledgements

I would like to express gratitude to my supervisor, Prof. Peter Y. A. Ryan, for his kindness and advice throughout the course of my Ph.D, and to my co-supervisor, Dr. Qiang Tang, for his continuous support and for suggesting me the topic of this thesis, at a time when the term "function privacy" had not been coined yet. To you both, a warm thank you for allowing me to join APSIA group and making this thesis possible. To my fellow colleagues from APSIA, thank you for your friendship and for the many fun moments we had in the last four years.

My first steps into cryptography and research were guided by Prof. Manuel Barbosa and Dr. Pooya Farshim, during my Master's degree, and I had the great pleasure to keep working with them since. I could not thank you both enough for all you have taught me, for the enthusiastic discussions we had, and for the time and effort you dedicated to our research projects. I will always feel indebted to you.

I would also like to express my appreciation to Prof. Michel Abdalla for accepting being a member of my dissertation committee. His insightful comments were greatly valued.

Finally, I must acknowledge that my Ph.D was supported by Luxembourg National Research Fund (FNR), under AFR Grant No. 5107187, and in multiple occasions by the Doctoral School of Computer Science & Computer Engineering of the University of Luxembourg as well.

To my loving and supportive girlfriend, Joana de Castro Ribeiro, and to my always-encouraging parents, Alice Delerue and Carlos Arriaga.

Contents

List of Figures

Chapter 1

Introduction

As cloud services become increasingly popular, security concerns arise from exposing the user's data to third-party service providers. Encryption can be used to protect the user's data, but traditional public-key encryption is intrinsically designed to provide *all-or-nothing* security guarantees, meaning that either one possesses the decryption key and can decrypt the entire plaintext or nothing can be learned about the plaintext (besides, perhaps, its length). In this sense, usability is sacrificed if the owner of the data is unable to specify a decryption policy or delegate search operations to the storage service provider.

Over time, several cryptographic primitives emerged, with different use-case scenarios in mind, where one can learn a function of the plaintext. The first of such primitives was identity-based encryption (IBE), proposed by Shamir [Sha84] in 1984, but instantiations only appeared later in 2001 with the seminal papers of Boneh and Franklin [BF01] and Cocks [Coc01]. In an IBE scheme, an authority holding the master secret key can extract decryption keys associated with identities, which are bit-strings that uniquely identify an entity or a person, such as a passport number or an e-mail address. A ciphertext encrypted under the master public-key and (let's say) Alice's identity (e.g. "alice.liddell@uni.lu") can only be decrypted with Alice's decryption key, which only the holder of the master secret key can issue.

Other primitives provide search delegation functionalities. The simplest of wich is public-key encryption with keyword search (KS), introduced in [BDOP04]. It allows *exact-matching* searches to be carried out over ciphertexts. A typical scenario where this primitive can bring great benefits to users (and consequently to service providers wishing to increase their customer base as well) is that of any email system. Suppose user Alice stores her emails in the servers of some email service provider, so that she can access them from either her laptop or her smartphone. Alice does not trust the service provider or fears that government agencies may require the service provider to hand over all her data. Using standard public key encryption, any user with Alice's public key can send her encrypted emails that only she can decrypt. For Alice to find a particular email later on, the sender could also attach to the email some *searchable ciphertexts*, produced from a KS scheme, with keywords that Alice might use when searching for this email. These ciphertexts are searchable upon delegation, meaning that only Alice can authorize the email service provider to search on her behalf by issuing a token that encodes Alice's search criteria (e.g. ciphertexts that encrypt the keyword "project RAPID20130115"), generated from her own secret key. The service provider searches through all Alice's emails for those containing searchable ciphertexts that match the issued token, and returns to her only those with a positive match.

For more expressive search queries, the works of Boneh and Waters [BW07] and Katz, Sahai and Waters [KSW13] show how to compute conjunctive queries ($P_1 \wedge \ldots \wedge P_l$), disjunctive queries ($P_1 \vee \ldots \vee P_l$), subset queries ($x \in S$) and range queries ($x \in [a, b]$) from inner-product encryption (IPE) [AAB+15], where evaluating a token for vector $\mathbf{a}$ on a ciphertext that encrypts vector $\mathbf{b}$ gives 1 if and only if $\langle \mathbf{a}, \mathbf{b} \rangle = 0$. Some variants of IPE are defined such that the evaluation returns a payload message m [KSW13, LOS+10, AFV11], or the actual result of computing the inner product $\langle \mathbf{a}, \mathbf{b} \rangle$ [ABDP15, ABDP16]. The most recent developments IPE give rise to efficient schemes from standard assumptions.

More generally, the concept of functional encryption (FE) was independently formalized by Boneh, Sahai and Waters [BSW11] and O'Neil [O'N10]. In an FE scheme, the holder of the master secret key can issue tokens associated with functions of its choice. Possessing a token for function f allows one to recover $f(m)$, given an encryption of m. Informally, security dictates that only $f(m)$ is revealed about m and nothing else. One can easily see that IBE, KS and IPE fall in as particular cases of functional encryption with restricted functionality support. The most common approach is to model functions as circuits, and in an important fundamental result, Garg et al. [GGH+13] put forth the first construction of an FE scheme supporting all polynomial-size circuits based on indistinguishability obfuscation (iO) for circuits, which is now known as a central hub for the realization of many cryptographic primitives [SW14].

Standard notions of security for functional encryption [BSW11, O'N10] do not cover important use cases where, not only encrypted data, but also the functions associated with decryption tokens contain sensitive information. They guarantee that nothing about the plaintexts beyond query results are revealed to the server. However, they do not guarantee that the performed query, which may also contain sensitive information, remains hidden, which is of particular relevance in the context of searchable encryption. *Function privacy* is an emerging new notion that aims to address this problem. The difficulty of formalizing it lies in the evaluation functionality of FE, as the holder of a token for f may encrypt arbitrary messages using the public key, and obtain a large number of evaluations of f via the decryption algorithm. This means that function privacy can only be achieved as long as the token holder is unable to learn f through such an attack. How to define function privacy and how construct FE schemes that offer such security guarantees are the main questions addressed in this thesis.

1.1 Related work

The formal study of function privacy begins in the work of Boneh, Raghunathan and Segev [BRS13a], where the authors focused on identity-based encryption (IBE) and presented the first constructions offering various degrees of privacy. From the onset, it became clear that formalizing such a notion is challenging, even for simple functionalities such as IBE, as a large number of evaluation can always be computed for each token via the decryption algorithm. Boneh et al. therefore considered privacy for identities with high min-entropy. In general, however, the previous observation implies that function privacy can only be achieved as long as the token holder is unable to learn the function associated with the token through function evaluation, immediately suggesting a strong connection between private functional encryption and obfuscation.

Boneh, Raghunathan and Segev [BRS13a, BRS13b] give indistinguishability-based definitions of function privacy for IBE and subspace membership (a generalization of inner-product encryption). Roughly speaking, the IBE model imposes that whenever the token queries of the adversary have high min-entropy (or form a block source), decryption tokens will be indistinguishable from those corresponding to identities sampled from the uniform distribution. For subspace membership, the definition requires the random variables associated with vector components to be a block source.

Tokens for high-entropy identities, however, rarely exist in isolation and are often available in conjunction with ciphertexts encrypted for the very same identities. To address this requirement, the same authors [BRS13a] proposed an *enhanced* model for IBE in which the adversary also gets access to ciphertexts encrypted for identities associated with the challenge tokens. We show this model to be infeasible under the formalism of Boneh et al., as correlations with encrypted identities can lead to distinguishing attacks, e.g. via *repetition patterns.* (We will discuss this later in the thesis.) Although the model can be salvaged by further restricting the class of admissible distributions, it becomes primitive-specific and formulating a definition for other functionalities is not obvious (and indeed a similar extension was not formalized for subspace membership in [BRS13b]). Additionally, this model also falls short of capturing *arbitrary* correlations between encrypted messages and tokens, as it does not allow an adversary to see ciphertexts for identities which, although correlated with those extracted in the challenge tokens, *do not match any of them.*

Recently, Agrawal et al. [AAB+15] put forth a model for functional encryption that aims to address this problem with a very general UC-style definition (called "wishful security"). The core of the definition is an ideal security notion for functional encryption, which makes it explicit that both data privacy and function privacy should be simultaneously enforced. However, not only is this general simulation-based definition difficult to work with, but also aiming for it would amount to constructing virtual black-box obfuscation, for which strong impossibility results are known [BGI+01, GK05]. Indeed, the positive results of [AAB+15] are obtained in idealized models of computation.

1.2 Our contributions

The above discussion highlights the need for a general and convenient definition of privacy that incorporates arbitrary correlations between decryption tokens and encrypted messages, and yet can be shown to be feasible without relying on idealized models of computation. The first contribution of our work is an *indistinguishability-based* definition that precisely models arbitrary correlations for general circuits. Our definition builds on a framework for *unpredictable* samplers and unifies within a single definition all previous indistinguishability-based notions.

The second contribution of this thesis is five constructions of private functional encryption schemes supporting different classes of functions and meeting varying degrees of security: (1) a white-box construction of an Anonymous IBE scheme based on composite-order groups, shown to be secure in the absence of correlated messages; (2) a simple and functionality-agnostic black-box construction from obfuscation, also shown to be secure in the absence of correlated messages; (3) a more evolved and still functionality-agnostic construction that achieves a form of function privacy that tolerates limited correlations between messages and functions; (4) a KS scheme achieving privacy in the presence of correlated messages beyond all previously proposed indistinguishability-based security definitions; (5) a KS construction that achieves our strongest notion of privacy (but relies on a more expressive form of obfuscation than the previous construction). We also develop an obfuscator for hyperplane membership that, when plugged into the third construction above gives rise to a private inner-product encryption scheme, answering a question left open by Boneh, Raghunathan and Segev [BRS13b] on how to define and realize *enhanced* security (i.e., privacy in the presence of correlated messages) for schemes supporting this functionality.

The standard approach in FE is to model complex functions as circuits, which yields inefficient evaluations over large inputs. As our third contribution, we propose a new primitive that we call "updatable functional encryption" (UFE), where instead of circuits we deal with RAM programs, which are closer to how programs are expressed in von Neumann architecture. We impose strict efficiency constrains and we envision tokens that are capable of updating the ciphertext, over which other tokens can be subsequently executed. We define a security notion for our primitive and propose a candidate construction from obfuscation, which serves as a starting point towards the realization of other schemes and contributes to the study on how to compute RAM programs over public-key encrypted data.

The unpredictability framework. At the core of our definitional work lies a precise definition characterizing which distributions over circuits and what correlated side information can be tolerated by a private FE scheme. We build on ideas from obfuscation [BST14, BC14, BBC$^+$14], functional encryption [BSW11, O'N10] and prior work in function privacy [BRS13a, BRS13b, ATR14, AAB$^+$15] to define a game-based notion of *unpredictability* for general functions. Our definition allows a *sampler* $\mathcal{S}$ to output a pair of circuit vectors $(\mathbf{C}_0, \mathbf{C}_1)$ and a pair of message vectors $(\mathbf{m}_0, \mathbf{m}_1)$ with arbitrary correlations between them, along with some side information z. Unpredictability then imposes that no predictor $\mathcal{P}$ interacting with oracles computing evaluations on these circuits and messages can find a point x such that $\mathbf{C}_0(x) \neq \mathbf{C}_1(x)$. (We do not impose indistinguishability, which is stronger, results in a smaller class of unpredictable samplers, and hence leads to weaker security.) The predictor $\mathcal{P}$ sees z and the outputs of the sampled circuits on the sampled messages. It can run in bounded or unbounded time, but it can only make polynomially many oracle queries to obtain additional information about the sampled circuits and messages. To avoid attacks that arise in the presence of computationally unpredictable auxiliary information [BM14, BST16] we adopt *unbounded* prediction later in the security analysis of our private functional encryption schemes.

This formalism fixes the unpredictability notion throughout the thesis. We can then capture specific types of samplers by imposing extra structural requirements on them. For instance, we may require the sampler to output a bounded number of circuits and messages, or include specific data in the auxiliary information, or not include any auxiliary information at all. Imposing that the sampler outputs single-circuit vectors, no messages, and includes the circuits as auxiliary information leads to the notion of *differing-inputs obfuscation* [ABG$^+$13, BST14]. Further imposing that the sampler also includes in the auxiliary information its random coins or allowing the predictor to run in unbounded time leads to *public-coin differing-inputs obfuscation* [IPS15] and *indistinguishability obfuscation* [GR14, BM14, GGH$^+$13], respectively. A sampler outputting circuits *and* messages comes to hand to model the privacy for functional encryption. We emphasize that our definition intentionally does *not* require the messages to be unpredictable. Further discussion on this choice can be found in Chapter 3.

THE PRIV MODEL. Building on unpredictability, we put forth a new indistinguishability-based notion of function privacy. Our notion, which we call PRIV, bears close resemblance to the standard IND-CPA model for functional encryption: it comes with a left-or-right LR oracle, a token-extraction TGEN oracle and the goal of the adversary is to guess a bit. The power of the model lies in that we endow LR with the ability to generate arbitrary messages and circuits via an unpredictable sampler. Trivial attacks are excluded by the joint action of unpredictability and the usual FE legitimacy condition, imposing equality of images on left and right. The enhanced model of Boneh, Raghunathan and Segev [BRS13a] falls in as a special case where the sampler is structurally restricted to be a block source. But our definition goes well beyond this and considers arbitrary and possibly low-entropy correlations. Furthermore, since unpredictability is *not* imposed on messages, PRIV implies IND-CPA security, and consequently it also guarantees *anonymity* for primitives such as IBE and ABE [BSW11]. Correlated circuits may be "low entropy" as long as they are identical on left and right, and since previous definitions adopted a real-or-random definition, they had to exclude this possibility. By giving the sampler the option to omit, manipulate and repeat the messages, our security notion implies previous indistinguishability-based notions in the literature, including those in [BRS13a, BRS13b, ATR14, AAB+15].

The implications of our new definition become clearer when we focus on (public-key encryption with) keyword search (KS) [BDOP04]. Consider a scenario where a client searches for a keyword but obtains no matching ciphertexts. The client then slightly modifies the keyword and requests a new search, now resulting in one or more successful matches. In this setting, the server sees ciphertexts encrypting unknown keywords that are closely related to keywords which the server holds tokens for. Our model ensures that if searched keywords are unpredictable from the perspective of the server, this uncertainty is preserved by the KS scheme after the searches are carried out. This does *not* imply that the server will be unable to distinguish a sequence of successful queries over the *same* high-entropy keyword, from a sequence of successful queries over *different* high-entropy keywords (this is impossible to achieve [ATR14]). However, when keyword searches do *not* match any of the correlated ciphertexts, then search patterns are guaranteed to remain hidden, even in the presence of low-entropy correlated encrypted keywords. We note that this captures a strong notion of unlinkability and untraceability between *unmatched queries*.

Constructions. We start by looking at Boyen and Waters [BW06] anonymous identity-based encryption scheme in the hope of showing that it already achieves some form of function privacy, as the decryption keys are randomized. Towards this end, we first present a simplified version of the original scheme and show that, in the random oracle model, not only IND-CPA security is still guaranteed, we are also able to lift the selective-id constraint in the proof. Next, we show that the scheme is PRIV secure up to two decryption keys, in the absence of correlated messages. In fact, we also show that if the sampler outputs vectors with just three identities, there is a trivial attack. To improve security, we extend the scheme to groups of composite order and show that the extended version is secure for an unbounded number of keys.

Taking a more general approach, we then formalize the intuition that obfuscating circuits before extraction should provide some level of privacy in FE. Using unpredictable samplers, we first generalize distributionally-indistinguishable (DI) obfuscators [BC14] from point functions to general circuits. Our obfuscate-then-extract OX transform shows that PRIV security in the absence of correlated messages can be achieved using DI obfuscators. In the reverse direction, we also established that some weak form of DI obfuscation (for samplers outputting single-circuit vectors) is also necessary. We also show that *composable* VGB obfuscation implies full-fledged DI obfuscation. So, emerging positive results on composable VGB obfuscation [BCKP14, BC14] already lead to PRIV-secure functional encryption schemes (supporting the same class of circuits as the obfuscator) in the absence of correlated messages.

To move beyond the above *token-only* model, we need to "decouple" the correlations between encrypted messages and challenge circuits so we can take advantage of FE security (that protects ciphertexts) and obfuscation (that protects the circuits) in a cumulative way. Building on ideas from [ABSV15] and [BCKP14] we identify a class of *concentrated* samplers that can be used in conjunction with the so-called "trojan" method—a technique to boost selective security to adaptive security in FE—to achieve function privacy. This construction improves on the security guarantees of OX considerably, but comes with the caveat that a mild restriction on second-stage token queries must be imposed: they must reveal (via circuit outputs) no more information about encrypted correlated messages than those revealed by first-stage queries. We give non-trivial examples of concentrated samplers and derive constructions for classes of circuits that encompass, among other functionalities, IBE, KS and *inner-product encryption*. By giving a construction of a DI obfuscator for hyperplane membership, we resolve a question left open by Boneh, Raghunathan and Segev [BRS13b] on the extension and realization of *enhanced security* for inner-product encryption.

Our forth construction is specific to point functions, and besides being simpler and more efficient, can tolerate arbitrary correlations between challenge keywords and encrypted messages. Put differently this construction removes the concentration restriction on samplers. For this construction we require a functional encryption scheme that supports the OR composition of two DI-secure point obfuscations. The composable VGB point obfuscator of Bitansky and Canetti [BC14] implies that the required DI point obfuscator exists. Furthermore, we also rely on a standard functional encryption scheme that supports the evaluations of four group operations in a DDH group (corresponding to the disjunction of two point function obfuscations), which is a relatively modest computation. We are, however, unable to lift the mild second-stage restriction.

Our last construction lifts the second-stage restriction at the cost of relying on more expressive forms of obfuscators. The novelty in this construction resides in the observation that, in order to offer the keyword search functionality, it suffices to encrypt information that enables equality checks between words and messages to be carried out. In our fourth construction we encode a message m as an *obfuscation* of the point function $\mathsf{C}[\mathsf{m}]$. Concretely, we obfuscate words before extraction *and* messages before encryption. Equality with w can be checked using a circuit $\mathsf{D}[\mathsf{w}]$ that on input an obfuscated point function $\mathsf{Obf}(\mathsf{C}[\mathsf{m}])$ returns $\mathsf{Obf}(\mathsf{C}[\mathsf{m}])(\mathsf{w})$. We emphasize that $\mathsf{D}[\mathsf{w}]$ is *not* a point function. We also need to ensure that an attacker cannot exploit the $\mathsf{D}[\mathsf{w}]$ circuits by, say, encrypting obfuscations of malicious circuits of its choice. We do this using NIZK proofs to ensure the outputs of the point obfuscator are *verifiable*: one can publicly verify that an obfuscation indeed corresponds to some point function. To summarize, our construction relies on a DI obfuscator supporting point functions $\mathsf{C}[\mathsf{m}](\mathsf{w}) := (\mathsf{m} = \mathsf{w})$ and circuits $\mathsf{D}[\mathsf{w}](\overline{\mathsf{C}}) := \overline{\mathsf{C}}(\mathsf{w})$ and a general-purpose FE. The circuits $\mathsf{C}[\mathsf{m}]$ and $\mathsf{D}[\mathsf{w}]$ were used negatively by Barak et al. [BGI+01] to launch generic attacks against VBB. Here, the restrictions imposed on legitimate PRIV samplers ensure that these attacks cannot be carried out in our setting, and obfuscators supporting them can be used positively to build private FE schemes.

Chapter 2

Preliminaries, Notation and Standard Definitions

2.1 Notation

We denote the security parameter by $\lambda \in \mathbb{N}$ and assume it is implicitly given to all algorithms in unary representation 1^λ. We denote the set of all bit strings of length ℓ by $\{0,1\}^\ell$ and the length of a string x by $|x|$. The bit complement of a string x is denoted by $\overline{x}$. We use the symbol ε to denote the empty string. A vector of strings $\mathbf{x}$ is written in boldface, and $\mathbf{x}[i]$ denotes its ith entry. The number of entries of $\mathbf{x}$ is denoted by $|\mathbf{x}|$. For a finite set X, we denote its cardinality by $|X|$ and the action of sampling a uniformly random element x from X by $x \leftarrow_{\$} X$. For a random variable X we denote its support by $[X]$. For a circuit C we denote its size by $|\mathsf{C}|$. We call a real-valued function $\mu(\lambda)$ negligible if $\mu(\lambda) \in \mathcal{O}(\lambda^{-\omega(1)})$ and denote the set of all negligible functions by NEGL. Throughput this thesis, $\perp$ denotes a special failure symbol outside the spaces underlying a cryptographic primitive. We adopt the code-based game-playing framework [BR06]. As usual "ppt" stands for probabilistic polynomial time.

CIRCUIT FAMILIES. Let $\mathsf{MSp} := \{\mathsf{MSp}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathsf{OSp} := \{\mathsf{OSp}_\lambda\}_{\lambda\in\mathbb{N}}$ be two families of finite sets parametrized by a security parameter $\lambda \in \mathbb{N}$. A circuit family $\mathsf{CSp} := \{\mathsf{CSp}_\lambda\}_{\lambda\in\mathbb{N}}$ is a sequence of circuit sets indexed by the security parameter. We assume that for all $\lambda \in \mathbb{N}$, all circuits in CSp_λ share a common input domain MSp_λ and output space OSp_λ. We also assume that membership in sets can be efficiently decided. For a vector of circuits $\mathbf{C} = [\mathsf{C}_1, \ldots, \mathsf{C}_n]$ and a vector of messages $\mathbf{m} = [\mathsf{m}_1, \ldots, \mathsf{m}_m]$ we define $\mathbf{C}(\mathbf{m})$ to be an $n \times m$ matrix whose ijth entry is $\mathsf{C}_i(\mathsf{m}_j)$. When $\mathsf{OSp}_\lambda = \{0, 1\}$ for all values of λ we call the circuit family Boolean.

Trees. We associate a tree T with the set of its nodes $\{\mathsf{node}_{i,j}\}$. Each node is indexed by a pair of non-negative integers representing the position (level and branch) of the node on the tree. The root of the tree is indexed by $(0, 0)$, its children have indices $(1, 0)$, $(1, 1)$, etc. A binary tree is *perfectly balanced* if every leaf is at the same level.

2.2 Bilinear groups and complexity assumptions

We first review the basic properties of prime-order bilinear groups and the computational assumptions *Decision Bilinear Diffie-Hellman* (DBDH) [BF01] and *Decision Linear* (DLIN) [BBS04]. We then review the properties of composite-order bilinear groups and the *Composite 3-party Diffie-Hellman* (C3DH) assumption made in [BW07]. Looking ahead, these properties are relevant for our white-box construction of a function-private AIBE scheme, described in Section 6.1.

2.2.1 Bilinear groups of prime order

A prime-order bilinear group generator is an algorithm $\mathcal{G}_\mathcal{P}$ that takes as input a security parameter 1^λ and outputs a description $\Gamma = (\mathsf{p}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g})$ where:

- $\mathbb{G}$ and $\mathbb{G}_\mathsf{T}$ are groups of order p with efficiently-computable group laws, where p is a λ-bit prime.
- g is a generator of $\mathbb{G}$.
- $\mathbf{e}$ is an efficiently-computable bilinear pairing $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_\mathsf{T}$, i.e. a map satisfying the following properties:
  - Bilinearity: $\forall \mathsf{a}, \mathsf{b} \in \mathbb{Z}_\mathsf{p}, \mathbf{e}(\mathsf{g}^\mathsf{a}, \mathsf{g}^\mathsf{b}) = \mathbf{e}(\mathsf{g}, \mathsf{g})^{\mathsf{ab}}$;
  - Non-degeneracy: $\mathbf{e}(\mathsf{g}, \mathsf{g}) \neq 1$.

Definition 1. *We say the DBDH assumption holds for group generator $\mathcal{G}_{\mathcal{P}}$ if for every ppt adversary $\mathcal{A}$ we have that*

$$\mathbf{Adv}^{\mathsf{dbdh}}_{\mathcal{G}_{\mathcal{P}},\mathcal{A}}(\lambda) := 2 \cdot \Pr[\mathsf{DBDH}^{\mathcal{A}}_{\mathcal{G}_{\mathcal{P}}}(1^{\lambda})] - 1 \in \textsc{Negl},$$

where game DBDH *is described in Figure 2.1 on the left.*

Definition 2. *We say the DLIN assumption holds for group generator $\mathcal{G}_{\mathcal{P}}$ if for every ppt adversary $\mathcal{A}$ we have that*

$$\mathbf{Adv}^{\mathsf{dlin}}_{\mathcal{G}_{\mathcal{P}},\mathcal{A}}(\lambda) := 2 \cdot \Pr[\mathsf{DLIN}^{\mathcal{A}}_{\mathcal{G}_{\mathcal{P}}}(1^{\lambda})] - 1 \in \textsc{Negl},$$

where game DLIN *is described in Figure 2.1 on the right.*

```
$\underline{\mathrm{DBDH}^{\mathcal{A}}_{\mathcal{G}_{\mathcal{P}}}(1^{\lambda})}$:
$\Gamma \leftarrow_{\$} \mathcal{G}_{\mathcal{P}}(1^{\lambda})$
$(\mathsf{p}, \mathbb{G}, \mathbb{G}_{\mathsf{T}}, \mathbf{e}, \mathsf{g}) \leftarrow \Gamma$
$\mathsf{z}_1, \mathsf{z}_2, \mathsf{z}_3 \leftarrow_{\$} \mathbb{Z}_{\mathsf{p}}$
$b \leftarrow_{\$} \{0,1\}$
if $(b = 0)$ then $\mathsf{Z} \leftarrow \mathbf{e}(\mathsf{g},\mathsf{g})^{\mathsf{z}_1\mathsf{z}_2\mathsf{z}_3}$
else $\mathsf{Z} \leftarrow_{\$} \mathbb{G}_{\mathsf{T}}$
$b' \leftarrow_{\$} \mathcal{A}(\Gamma, \mathsf{g}^{\mathsf{z}_1}, \mathsf{g}^{\mathsf{z}_2}, \mathsf{g}^{\mathsf{z}_3}, \mathsf{Z})$
return $(b = b')$
```

```
$\underline{\mathrm{DLIN}^{\mathcal{A}}_{\mathcal{G}_{\mathcal{P}}}(1^{\lambda})}$:
$\Gamma \leftarrow_{\$} \mathcal{G}_{\mathcal{P}}(1^{\lambda})$
$(\mathsf{p}, \mathbb{G}, \mathbb{G}_{\mathsf{T}}, \mathbf{e}, \mathsf{g}) \leftarrow \Gamma$
$\mathsf{z}_1, \mathsf{z}_2, \mathsf{z}_3, \mathsf{z}_4 \leftarrow_{\$} \mathbb{Z}_{\mathsf{p}}$
$b \leftarrow_{\$} \{0,1\}$
if $(b = 0)$ then $\mathsf{Z} \leftarrow \mathsf{g}^{\mathsf{z}_3+\mathsf{z}_4}$
else $\mathsf{Z} \leftarrow_{\$} \mathbb{G}_{\mathsf{T}}$
$b' \leftarrow_{\$} \mathcal{A}(\Gamma, \mathsf{g}^{\mathsf{z}_1}, \mathsf{g}^{\mathsf{z}_2}, \mathsf{g}^{\mathsf{z}_1\mathsf{z}_3}, \mathsf{g}^{\mathsf{z}_2\mathsf{z}_4}, \mathsf{Z})$
return $(b = b')$
```

Figure 2.1: Games defining DBDH and DLIN computational assumptions.

2.2.2 Bilinear groups of composite order

A composite-order bilinear group generator is an algorithm $\mathcal{G}_{\mathcal{C}}$ that takes as input a security parameter 1^{λ} and outputs a description $\Gamma = (\mathsf{p}, \mathsf{q}, \mathbb{G}, \mathbb{G}_{\mathsf{T}}, \mathbf{e}, \mathsf{g})$ where:

- $\mathbb{G}$ and $\mathbb{G}_{\mathsf{T}}$ are groups of order $\mathsf{n} = \mathsf{pq}$, where p and q are independent λ-bit primes, with efficiently computable group laws.
- g is a generator of $\mathbb{G}$.
- $\mathbf{e}$ is an efficiently-computable bilinear pairing $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathsf{T}}$, i.e. a map satisfying the following properties:

– Bilinearity: $\forall \mathsf{a}, \mathsf{b} \in \mathbb{Z}_\mathsf{n}, \mathbf{e}(\mathsf{g}^\mathsf{a}, \mathsf{g}^\mathsf{b}) = \mathbf{e}(\mathsf{g}, \mathsf{g})^{\mathsf{ab}}$;

– Non-degeneracy: $\mathbf{e}(\mathsf{g}, \mathsf{g}) \neq 1$.

Subgroups $\mathbb{G}_\mathsf{p} \subset \mathbb{G}$ and $\mathbb{G}_\mathsf{q} \subset \mathbb{G}$ of order p and order q can be generated respectively by $\mathsf{g}_\mathsf{p} = \mathsf{g}^\mathsf{q}$ and $\mathsf{g}_\mathsf{q} = \mathsf{g}^\mathsf{p}$. We recall some important properties regarding these groups:

- $\mathbb{G} = \mathbb{G}_\mathsf{p} \times \mathbb{G}_\mathsf{q}$
- $\mathbf{e}(\mathsf{g}_\mathsf{p}, \mathsf{g}_\mathsf{q}) = \mathbf{e}(\mathsf{g}^\mathsf{q}, \mathsf{g}^\mathsf{p}) = \mathbf{e}(\mathsf{g}, \mathsf{g})^\mathsf{n} = 1$
- $\mathbf{e}(\mathsf{g}_\mathsf{p}, (\mathsf{g}_\mathsf{p})^\mathsf{a} \cdot (\mathsf{g}_\mathsf{q})^\mathsf{b}) = \mathbf{e}(\mathsf{g}_\mathsf{p}, (\mathsf{g}_\mathsf{p})^\mathsf{a}) \cdot \mathbf{e}(\mathsf{g}_\mathsf{p}, (\mathsf{g}_\mathsf{q})^\mathsf{b}) = \mathbf{e}(\mathsf{g}_\mathsf{p}, \mathsf{g}_\mathsf{p})^\mathsf{a}$

Definition 3. *We say the C3DH assumption holds for group generator $\mathcal{G}_\mathcal{C}$ if for every ppt adversary $\mathcal{A}$ we have that*

$$\mathbf{Adv}^{\mathsf{c3dh}}_{\mathcal{G}_\mathcal{C},\mathcal{A}}(\lambda) := 2 \cdot \Pr[\mathsf{C3DH}^{\mathcal{A}}_{\mathcal{G}_\mathcal{C}}(1^\lambda)] - 1 \in \textsc{Negl},$$

where game C3DH *is described in Figure 2.2.*

```
$\underline{\mathrm{C3DH}^{\mathcal{A}}_{\mathcal{G}_\mathcal{C}}(1^\lambda)\text{:}}$
$(\mathsf{p}, \mathsf{q}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}) \leftarrow_\$ \mathcal{G}_\mathcal{C}(1^\lambda)$
$\mathsf{n} \leftarrow \mathsf{pq}; \quad \mathsf{g}_\mathsf{p} \leftarrow \mathsf{g}^\mathsf{q}; \quad \mathsf{g}_\mathsf{q} \leftarrow \mathsf{g}^\mathsf{p}$
$\Gamma' \leftarrow (\mathsf{n}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}, \mathsf{g}_\mathsf{p}, \mathsf{g}_\mathsf{q})$
$\mathsf{X}_1, \mathsf{X}_2, \mathsf{X}_3 \leftarrow_\$ \mathbb{G}_\mathsf{q}$
$\mathsf{a}, \mathsf{b}, \mathsf{c} \leftarrow_\$ \mathbb{Z}_\mathsf{n}$
$b \leftarrow_\$ \{0, 1\}$
if $(b = 0)$ then $\mathsf{R} \leftarrow \mathsf{X}_3(\mathsf{g}_\mathsf{p})^\mathsf{c}$
else $\mathsf{R} \leftarrow_\$ \mathbb{G}$
$b' \leftarrow_\$ \mathcal{A}(\Gamma', (\mathsf{g}_\mathsf{p})^\mathsf{a}, (\mathsf{g}_\mathsf{p})^\mathsf{b}, \mathsf{X}_1(\mathsf{g}_\mathsf{p})^{\mathsf{ab}}, \mathsf{X}_2(\mathsf{g}_\mathsf{p})^{\mathsf{abc}}, \mathsf{R})$
return $(b = b')$
```

Figure 2.2: Game defining C3DH computational assumption.

2.3 Cryptographic primitives

We now review standard definitions of several cryptographic primitives, namely pseudorandom permutations, public-key encryption, functional encryption (and its particular forms anonymous identity-based encryption, keyword search encryption and inner-product encryption), non-interactive zero-knowledge proof systems, collision-resistant hash functions and puncturable pseudorandom functions. The definitions of functional encryption will serve as a starting point towards the security modelling of what we call *private functional encryption*. The remaining cryptographic primitives will serve as building blocks in our constructions.

2.3.1 Pseudorandom permutations

Let $\mathsf{KSp} := \{\mathsf{KSp}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathsf{MSp} := \{\mathsf{MSp}_\lambda\}_{\lambda\in\mathbb{N}}$ be two families of finite sets parametrized by a security parameter $\lambda \in \mathbb{N}$. A pseudorandom permutation (PRP) family $\mathsf{PRP} := (\mathsf{K}, \mathsf{E}, \mathsf{D})$ is a triple of ppt algorithms as follows. (1) K on input the security parameter outputs a uniform element in KSp_λ; (2) E is deterministic and on input a key $\mathsf{k} \in \mathsf{KSp}_\lambda$ and a point $\mathsf{x} \in \mathsf{MSp}_\lambda$ outputs a point in MSp_λ; (3) D is deterministic and on input a $\mathsf{k} \in \mathsf{KSp}_\lambda$ and a point $\mathsf{x} \in \mathsf{MSp}_\lambda$ outputs a point in MSp_λ. The PRP family PRP is correct if for all $\lambda \in \mathbb{N}$, all $\mathsf{k} \in \mathsf{KSp}_\lambda$ and all $\mathsf{x} \in \mathsf{MSp}_\lambda$ we have that $\mathsf{D}(\mathsf{k}, \mathsf{E}(\mathsf{k}, \mathsf{x})) = \mathsf{x}$. A pseudorandom permutation $\mathsf{PRP} := (\mathsf{K}, \mathsf{E}, \mathsf{D})$ is called PRP secure if for every ppt adversary $\mathcal{A}$ we have that

$$\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{PRP},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathrm{PRP}^{\mathcal{A}}_{\mathsf{PRP}}(1^\lambda)\right] - 1 \in \textsc{Negl}$$

where game $\mathrm{PRP}^{\mathcal{A}}_{\mathsf{PRP}}(1^\lambda)$ is defined in Figure 2.3. For our purposes, we rely on the non-strong security notion where inverse queries are not allowed. Furthermore, we do not necessarily require the inverse map D to be efficiently computable.

```
$\mathrm{PRP}^{\mathcal{A}}_{\mathsf{PRP}}(1^\lambda)$:
$b \leftarrow_\$ \{0,1\}$
$\mathsf{k} \leftarrow_\$ \mathsf{K}(1^\lambda)$
$b' \leftarrow_\$ \mathcal{A}^{\textsc{Fn}}(1^\lambda)$
return $(b = b')$

$\textsc{Fn}(\mathsf{x})$:
if $T[\mathsf{x}] = \perp$ then
    $T[\mathsf{x}] \leftarrow_\$ \mathsf{MSp}_\lambda \setminus T$
if $b = 1$ return $T[\mathsf{x}]$
else return $\mathsf{E}(\mathsf{k}, \mathsf{x})$
```

Figure 2.3: Game defining the security of a pseudorandom permutation PRP.

2.3.2 Public-key encryption

A public-key encryption scheme $\mathsf{PKE} := (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ with message space $\mathsf{MSp} := \{\mathsf{MSp}_\lambda\}_{\lambda\in\mathbb{N}}$ and randomness space $\mathsf{RSp} := \{\mathsf{RSp}_\lambda\}_{\lambda\in\mathbb{N}}$ is specified by three ppt algorithms as follows. (1) $\mathsf{PKE.Setup}(1^\lambda)$ is the probabilistic key-generation algorithm, taking as input the security parameter and returning a secret key sk and a public key pk. (2) $\mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{m}; \mathsf{r})$ is the probabilistic encryption algorithm. On input a public key pk, a message $\mathsf{m} \in \mathsf{MSp}_\lambda$ and possibly some random coins $\mathsf{r} \in \mathsf{RSp}_\lambda$, this algorithm outputs a ciphertext c. (3) $\mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{c})$ is the deterministic decryption algorithm. On input of a secret key sk and a ciphertext c, this algorithm outputs a message $\mathsf{m} \in \mathsf{MSp}_\lambda$ or failure symbol $\perp$.

Correctness. The correctness of a public-key encryption scheme requires that for any $\lambda \in \mathbb{N}$, any $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{PKE.Setup}(1^\lambda)]$, any $\mathsf{m} \in \mathsf{MSp}_\lambda$ and any random coins $\mathsf{r} \in \mathsf{RSp}_\lambda$, we have that $\mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{m}; \mathsf{r})) = \mathsf{m}$.

Security. We recall the standard security notions of *indistinguishability under chosen ciphertext attacks* (IND-CCA) and its weaker variant known as *indistinguishability under chosen plaintext attacks* (IND-CPA). We say that a public-key encryption scheme PKE is IND-CCA secure if for every *legitimate* ppt adversary $\mathcal{A}$

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := 2 \cdot \Pr[\text{IND-CCA}^{\mathcal{A}}_{\mathsf{PKE}}(1^\lambda)] - 1\,,$$

where game $\text{IND-CCA}^{\mathcal{A}}_{\mathsf{PKE}}$ described in Figure 2.4, in which the adversary has access to a left-or-right challenge oracle (LR) and a decryption oracle (Dec). We say that $\mathcal{A}$ is legitimate if: (1) $|\mathsf{m}_0| = |\mathsf{m}_1|$ whenever the left-or-right oracle is queried; and (2) the adversary does not call the decryption oracle with $\mathsf{c} \in \mathsf{list}$. We obtain the weaker IND-CPA notion if the adversary is not allowed to place any decryption query.

```
$\underline{\text{IND-CCA}^{\mathcal{A}}_{\mathsf{PKE}}(1^\lambda)}$:
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow_{\$} \mathsf{PKE.Setup}(1^\lambda)$
    $b \leftarrow_{\$} \{0,1\}$
    $b' \leftarrow_{\$} \mathcal{A}^{\text{LR,Dec}}(1^\lambda, \mathsf{pk})$
    return $(b = b')$

$\underline{\text{LR}(\mathsf{m}_0, \mathsf{m}_1)}$:
    $\mathsf{c} \leftarrow_{\$} \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{m}_b)$
    $\mathsf{list} \leftarrow \mathsf{c} : \mathsf{list}$
    return $\mathsf{c}$

$\underline{\text{Dec}(\mathsf{c})}$:
    $\mathsf{m} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{c})$
    return $\mathsf{m}$
```

Figure 2.4: Game defining IND-CCA security of a public-key encryption scheme PKE.

2.3.3 Functional encryption

Syntax. A functional encryption scheme FE associated with a circuit family CSp is specified by four ppt algorithms as follows. (1) $\mathsf{FE.Gen}(1^\lambda)$ is the setup algorithm and on input a security parameter 1^λ it outputs a master secret key msk and a master public key mpk; (2) $\mathsf{FE.TGen}(\mathsf{msk}, \mathsf{C})$ is the token-generation algorithm and on input a master secret key msk and a circuit $\mathsf{C} \in \mathsf{CSp}_\lambda$ outputs a token tk for C; (3) $\mathsf{FE.Enc}(\mathsf{mpk}, \mathsf{m})$ is the encryption algorithm and on input a master public key mpk and a message $\mathsf{m} \in \mathsf{MSp}_\lambda$ outputs a ciphertext c; (4) $\mathsf{FE.Eval}(\mathsf{c}, \mathsf{tk})$ is the deterministic evaluation (or decryption) algorithm and on input a ciphertext c and a token tk outputs a value $\mathsf{y} \in \mathsf{OSp}_\lambda$ or failure symbol $\perp$.

We adopt a *computational* notion of correctness for FE schemes and require that no ppt adversary is able to produce a message m and a circuit C that violates the standard correctness property of the FE scheme (that is, $\mathsf{FE.Eval}(\mathsf{FE.Enc}(\mathsf{mpk}, \mathsf{m}), \mathsf{FE.TGen}(\mathsf{msk}, \mathsf{C})) \neq \mathsf{C}(\mathsf{m})$), even with the help of an (unrestricted) token-generation oracle. We also adopt the standard notion of IND-CPA security [BSW11, O'N10] where an adversary with access to a token-generation oracle cannot distinguish encryptions of messages m_0, m_1 under the standard restriction that it cannot obtain a decryption token for a circuit C for which $\mathsf{C}(\mathsf{m}_0) \neq \mathsf{C}(\mathsf{m}_1)$.

Correctness. We will adopt a game-based definition of *computational* correctness for FE schemes which has been widely adopted in the literature [ABC+08, Gol04] and suffices for the overwhelming majority of use cases. Roughly speaking, this property requires that no efficient adversary is able to come up with a message and a circuit which violates the correctness property of the FE scheme, even with the help of an (unrestricted) token-generation oracle. Formally, we say that scheme FE is computationally correct if for all ppt adversaries $\mathcal{A}$

$$\mathbf{Adv}^{\mathrm{cc}}_{\mathsf{FE},\mathcal{A}}(\lambda) := \Pr\left[\mathrm{CC}^{\mathcal{A}}_{\mathsf{FE}}(1^\lambda)\right] \in \mathrm{NEGL} ,$$

where game $\mathrm{CC}^{\mathcal{A}}_{\mathsf{FE}}(1^\lambda)$ is shown in Figure 2.5 on the left. Perfect correctness corresponds to the setting where the above advantage is required to be zero.

Security. A functional encryption scheme FE is IND-CPA secure [BSW11, O'N10] if for any legitimate ppt adversary $\mathcal{A}$

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{FE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\text{IND-CPA}^{\mathcal{A}}_{\mathsf{FE}}(1^\lambda)\right] - 1 \in \text{NEGL} ,$$

where game IND-CPA$^{\mathcal{A}}_{\mathsf{FE}}(1^\lambda)$ is defined in Figure 2.5 on the right. We say $\mathcal{A}$ is legitimate if for all messages pairs queried to the left-or-right oracle, i.e., for all $(\mathsf{m}_0, \mathsf{m}_1) \in \mathsf{MList}$, and all extracted circuits $\mathsf{C} \in \mathsf{TList}$ we have that $\mathsf{C}(\mathsf{m}_0) = \mathsf{C}(\mathsf{m}_1)$.

```
$\underline{\text{CC}^{\mathcal{A}}_{\mathsf{FE}}(1^\lambda)}$:
  $(\mathsf{msk}, \mathsf{mpk}) \leftarrow_\$ \mathsf{FE.Gen}(1^\lambda)$
  $(\mathsf{m}, \mathsf{C}) \leftarrow_\$ \mathcal{A}^{\text{TGEN}}(\mathsf{mpk})$
  $\mathsf{c} \leftarrow_\$ \mathsf{FE.Enc}(\mathsf{mpk}, \mathsf{m})$
  $\mathsf{tk} \leftarrow_\$ \mathsf{FE.TGen}(\mathsf{msk}, \mathsf{C})$
  $y \leftarrow_\$ \mathsf{FE.Eval}(\mathsf{c}, \mathsf{tk})$
  return $(y \neq \mathsf{C}(\mathsf{m}))$

$\underline{\text{TGEN}(\mathsf{C})}$:
  $\mathsf{tk} \leftarrow_\$ \mathsf{FE.TGen}(\mathsf{msk}, \mathsf{C})$
  return $\mathsf{tk}$
```

```
$\underline{\text{IND-CPA}^{\mathcal{A}}_{\mathsf{FE}}(1^\lambda)}$:
  $(\mathsf{msk}, \mathsf{mpk}) \leftarrow_\$ \mathsf{FE.Gen}(1^\lambda)$
  $b \leftarrow_\$ \{0, 1\}$
  $b' \leftarrow_\$ \mathcal{A}^{\text{LR},\text{TGEN}}(\mathsf{mpk})$
  return $(b = b')$
```

```
$\underline{\text{LR}(\mathsf{m}_0, \mathsf{m}_1)}$:
  $\mathsf{c} \leftarrow_\$ \mathsf{FE.Enc}(\mathsf{mpk}, \mathsf{m}_b)$
  $\mathsf{MList} \leftarrow (\mathsf{m}_0, \mathsf{m}_1) : \mathsf{MList}$
  return $\mathsf{c}$

$\underline{\text{TGEN}(\mathsf{C})}$:
  $\mathsf{tk} \leftarrow_\$ \mathsf{FE.TGen}(\mathsf{msk}, \mathsf{C})$
  $\mathsf{TList} \leftarrow \mathsf{C} : \mathsf{TList}$
  return $\mathsf{tk}$
```

Figure 2.5: Games defining the computational correctness and IND-CPA security of a functional encryption scheme FE.

The IND-CPA notion self-composes in the sense that security against adversaries that place one LR query is equivalent to the setting where an arbitrary number of queries is allowed. It is also well known that IND-CPA security is weaker than generalizations of semantic security for functional encryption [BSW11, O'N10, BF13], and strong impossibility results for the latter have been established [BSW11, GVW12, AGVW13]. On the other hand, IND-CPA-secure FE schemes for all polynomial-size circuit families have been recently constructed [GVW12, GGH$^+$13, GKP$^+$13]. Other recent feasibility results have been established in weaker forms of the IND-CPA model such as the selective model [GVW12, GGH$^+$13, GKP$^+$13] where the adversary commits to its challenge messages at the onset; or the weak model for Boolean circuits, where the adversary is restricted to extract tokens that evaluate to 0 on the challenge messages [GVW15].

2.3.3.1 Anonymous identity-based encryption

Identity-based encryption (IBE) was first proposed by Shamir [Sha84], but instantiations only appeared later with the seminal papers of Boneh and Franklin [BF01] and Cocks [Coc01]. In an IBE scheme, an authority holding the master secret key can extract decryption keys associated with identities, such as passport numbers or an e-mail addresses. A ciphertext encrypted under the master public-key and some identity can only be decrypted with a decryption key for that identity, which only the authority holding the master secret key can issue. Here, we describe the syntax, correctness and security of *anonymous* identity-based encryption schemes (AIBE). Briefly, anonymity requires that the ciphertext also hides the identity for which it is intended.

Syntax. An anonymous identity based encryption scheme AIBE is a functional encryption scheme for a circuit family $\mathsf{CSp}_\lambda := \{\mathsf{C}[\mathsf{id}^\star] : \mathsf{id}^\star \in \mathsf{IdSp}_\lambda\}$, over identity space IdSp_λ and message space MSp_λ, such that each circuit C is defined as:

$$\mathsf{C}[\mathsf{id}^\star](\mathsf{id}, \mathsf{m}) := \begin{cases} \mathsf{m} & \text{if } (\mathsf{id} = \mathsf{id}^\star); \\ \perp & \text{otherwise.} \end{cases}$$

For simplicity, C is *canonically* represented by the identity $\mathsf{id}^\star$ with which it is associated. We write the algorithms of an AIBE scheme as follows. (1) $\mathsf{AIBE.Setup}(1^\lambda)$ is the setup algorithm and on input a security parameter 1^λ it outputs a master secret key msk and a master public key mpk; (2) $\mathsf{AIBE.Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{m})$ is the encryption algorithm and on input a master public key mpk, an identity $\mathsf{id} \in \mathsf{IdSp}_\lambda$ and a message $\mathsf{m} \in \mathsf{MSp}_\lambda$, it outputs a ciphertext c; (3) $\mathsf{AIBE.KeyGen}(\mathsf{msk}, \mathsf{id})$ is the key-generation algorithm and on input a master secret key msk and an identity $\mathsf{id} \in \mathsf{IdSp}_\lambda$ it outputs a decryption key sk_id; (4) finally, $\mathsf{AIBE.Dec}(\mathsf{sk}_\mathsf{id}, \mathsf{c})$ is the decryption algorithm that on input a secret key sk_id and a ciphertext c, it either outputs a message $\mathsf{m} \in \mathsf{MSp}_\lambda$ or a failure symbol $\perp$. The correctness and IND-CPA security of AIBE are defined identically to that of FE schemes supporting the circuit class described above. Note that this results in *semantic security* and *anonymity* in the traditional sense applied to identity-based encryption, i.e. ciphertexts conceal both the message and the identity of the recipient.

2.3.3.2 Keyword search encryption

Set circuits. In this work we are interested in Boolean circuits that assume the value 1 on only a polynomially large subset of their domains. We call these *set circuits.* We define the *canonical* representation of a set circuit C with its corresponding set S as the circuit $\mathsf{C}[S]$ that has the set S explicitly hardwired in it:

$$\mathsf{C}[S](\mathsf{m}) := \begin{cases} 1 & \text{if } \mathsf{m} \in S; \\ 0 & \text{otherwise.} \end{cases}$$

Formally, a family of Boolean circuits CSp is a set circuit family if there is a polynomial poly such that for all $\lambda \in \mathbb{N}$ and all $\mathsf{C} \in \mathsf{CSp}_\lambda$ we have that $|S(\mathsf{C})| \leq \mathsf{poly}(\lambda)$ where $S(\mathsf{C}) := \{\mathsf{m} \in \mathsf{MSp}_\lambda : \mathsf{C}(\mathsf{m}) = 1\}$. Point circuits/functions correspond to the case where $\mathsf{poly}(\lambda) = 1$. We use $\mathsf{C}[\mathsf{m}]$ to denote the point circuit that on input m returns 1 and 0 otherwise. Throughout the thesis, we assume that non-obfuscated set circuits are canonically represented.

Syntax. A public-key encryption with keyword search scheme (or simply a keyword search scheme) KS is a functional encryption scheme for a point circuit family over the message space: $\mathsf{CSp}_\lambda := \{\mathsf{C}[\mathsf{m}] : \mathsf{m} \in \mathsf{MSp}_\lambda\}$. We often identify circuit $\mathsf{C}[\mathsf{m}]$ with its message m, but in order to distinguish circuits from messages we use the term *keyword* to refer to the former. We write the algorithms associated to a KS scheme as $\mathsf{KS.Gen}(1^\lambda)$, $\mathsf{KS.Enc}(\mathsf{pk}, \mathsf{m})$, $\mathsf{KS.TGen}(\mathsf{sk}, \mathsf{w})$ and $\mathsf{KS.Test}(\mathsf{c}, \mathsf{tk})$, where the latter outputs either 0 or 1. The computational correctness of a KS scheme is defined identically to that of an FE scheme. We say the scheme has *no false negatives* if correctness advantage is negligible and, whenever $\mathcal{A}$ outputs $(\mathsf{w}, \mathsf{C}[\mathsf{w}])$, it is 0. IND-CPA security is also defined identically to FE schemes for point function families. Note that weak and standard IND-CPA notions are equivalent for KS schemes.

2.3.3.3 Inner-product encryption

Syntax. Let $\mathsf{CSp} := \{\mathsf{CSp}_p^d\}$ be a set circuit family of hyperplane membership testing functions that is defined for each value of the security parameter λ such that p is a λ-bit prime and d is a positive integer. Every circuit $\mathsf{C} \in \mathsf{CSp}_p^d$ is canonically represented by a vector $\vec{a} \in \mathbb{Z}_p^d$ (i.e. $\vec{a}$ is a vector of d elements from $\mathbb{Z}_p$) and returns 1 if and only if the input vector $\vec{x} \in \mathbb{Z}_p^d$ is *orthogonal* to $\vec{a}$. More precisely,

$$\mathsf{C}[\vec{a}](\vec{x}) := \begin{cases} 1 & \text{if } \langle \vec{x}, \vec{a} \rangle = 0; \\ 0 & \text{otherwise.} \end{cases}$$

An inner-product encryption scheme IPE is a functional encryption scheme for a hyperplane membership testing circuit family $\mathsf{CSp}_\lambda := \{\mathsf{C}[\vec{a}] : \vec{a} \in \mathbb{Z}_p^d\}$. The correctness and IND-CPA security of inner-product encryption are defined identically to that of FE schemes supporting the circuit class CSp as defined above.

2.3.4 NIZK proof systems

Syntax. A non-interactive zero-knowledge proof system for an **NP** language $\mathcal{L}$ with an efficiently computable binary relation $\mathcal{R}$ consists of three ppt algorithms as follows. (1) $\mathsf{NIZK.Setup}(1^\lambda)$ is the setup algorithm and on input a security parameter 1^λ it outputs a common reference string crs; (2) $\mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{x}, \omega)$ is the proving algorithm and on input a common reference string crs, a statement x and a witness ω it outputs a proof π or a failure symbol $\perp$; (3) $\mathsf{NIZK.Verify}(\mathsf{crs}, \mathsf{x}, \pi)$ is the verification algorithm and on input a common reference string crs, a statement x and a proof π it outputs either true or false.

Perfect completeness. Completeness imposes that an honest prover can always convince an honest verifier that a statement belongs to $\mathcal{L}$, provided that it holds a witness testifying to this fact. We say a NIZK proof is *perfectly complete* if for every (possibly unbounded) adversary $\mathcal{A}$

$$\mathbf{Adv}^{\text{complete}}_{\mathsf{NIZK},\mathcal{A}}(\lambda) := \Pr\left[\text{Complete}^{\mathcal{A}}_{\mathsf{NIZK}}(1^\lambda)\right] = 0 \ ,$$

where game $\text{Complete}^{\mathcal{A}}_{\mathsf{NIZK}}(1^\lambda)$ is shown in Figure 2.6 on the left.

Statistical soundness. Soundness imposes that a malicious prover cannot convince an honest verifier of a false statement. We say a NIZK proof is *perfectly sound* if for every (possibly unbounded) adversary $\mathcal{A}$ we have that

$$\mathbf{Adv}^{\text{sound}}_{\mathsf{NIZK},\mathcal{A}}(\lambda) := \Pr\left[\text{Sound}^{\mathcal{A}}_{\mathsf{NIZK}}(1^\lambda)\right] \in \text{Negl} \ ,$$

where game $\text{Sound}^{\mathcal{A}}_{\mathsf{NIZK}}(1^\lambda)$ is shown in Figure 2.6 on the right. If the above advantage is 0, we say the NIZK proof system is *perfectly sound*.

Computational zero knowledge. The zero-knowledge property guarantees that proofs do not leak information about the witnesses that originated them. Technically, this is formalized by requiring the existence of a ppt simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ where Sim_1 takes the security parameter 1^λ as input and outputs a simulated common reference string crs together with a trapdoor tp, and Sim_2 takes the trapdoor as input tp together with a statement $\mathsf{x} \in \mathcal{L}$ for which it must forge a proof π. We say a proof system is *computationally zero knowledge* if, for every ppt adversary $\mathcal{A}$, there exists a simulator Sim such that

$$\mathbf{Adv}^{\text{zk}}_{\mathsf{NIZK},\mathcal{A},\mathsf{Sim}}(\lambda) := \left|\Pr\left[\text{ZK-Real}^{\mathcal{A}}_{\mathsf{NIZK}}(1^\lambda)\right] - \left[\text{ZK-Ideal}^{\mathcal{A},\mathsf{Sim}}_{\mathsf{NIZK}}(1^\lambda)\right]\right| \in \textsc{Negl} \ ,$$

where games $\text{ZK-Real}^{\mathcal{A}}_{\mathsf{NIZK}}(1^\lambda)$ and $\text{ZK-Ideal}^{\mathcal{A},\mathsf{Sim}}_{\mathsf{NIZK}}(1^\lambda)$ are shown in Figure 2.7.

```
$\text{Complete}^{\mathcal{A}}_{\mathsf{NIZK}}(1^\lambda)$:
$\mathsf{crs} \leftarrow_\$ \mathsf{NIZK.Setup}(1^\lambda)$
$(\mathsf{x}, \omega) \leftarrow_\$ \mathcal{A}(1^\lambda, \mathsf{crs})$
if $(\mathsf{x}, \omega) \notin \mathcal{R}$ return 0
$\pi \leftarrow_\$ \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{x}, \omega)$
return $\neg(\mathsf{NIZK.Verify}(\mathsf{crs}, \mathsf{x}, \pi))$
```

```
$\text{Sound}^{\mathcal{A}}_{\mathsf{NIZK}}(1^\lambda)$:
$\mathsf{crs} \leftarrow_\$ \mathsf{NIZK.Setup}(1^\lambda)$
$(\mathsf{x}, \pi) \leftarrow_\$ \mathcal{A}(1^\lambda, \mathsf{crs})$
return $(\mathsf{x} \notin \mathcal{L} \wedge$
$\mathsf{NIZK.Verify}(\mathsf{crs}, \mathsf{x}, \pi))$
```

Figure 2.6: Games defining the completeness and soundness properties of a non-interactive zero-knowledge proof system NIZK.

```
$\text{ZK-Real}^{\mathcal{A}}_{\mathsf{NIZK}}(1^\lambda)$:
$\mathsf{crs} \leftarrow_\$ \mathsf{NIZK.Setup}(1^\lambda)$
$b \leftarrow_\$ \mathcal{A}^{\text{Prove}}(1^\lambda, \mathsf{crs})$

$\text{Prove}(\mathsf{x}, \omega)$:
if $(\mathsf{x}, \omega) \notin \mathcal{R}$ return $\perp$
$\pi \leftarrow_\$ \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{x}, \omega)$
return $\pi$
```

```
$\text{ZK-Ideal}^{\mathcal{A},\mathsf{Sim}}_{\mathsf{NIZK}}(1^\lambda)$:
$(\mathsf{crs}, \mathsf{tp}) \leftarrow_\$ \mathsf{Sim}_1(1^\lambda)$
$b \leftarrow_\$ \mathcal{A}^{\text{Prove}}(1^\lambda, \mathsf{crs})$

$\text{Prove}(\mathsf{x}, \omega)$:
if $(\mathsf{x}, \omega) \notin \mathcal{R}$ return $\perp$
$\pi \leftarrow_\$ \mathsf{Sim}_2(\mathsf{crs}, \mathsf{tp}, \mathsf{x})$
return $\pi$
```

Figure 2.7: Games defining the zero-knowledge property of a non-interactive zero-knowledge proof system NIZK.

2.3.5 Collision-resistant hash functions

A hash function family $\mathsf{H} := \{\mathsf{H}_\lambda\}_{\lambda\in\mathbb{N}}$ is a set parametrized by a security parameter $\lambda \in \mathbb{N}$, where each H_λ is a collection of functions mapping $\{0,1\}^m$ to $\{0,1\}^n$ such that $m > n$. The hash function family H is said to be collision-resistant if no ppt adversary $\mathcal{A}$ can find a pair of colliding inputs, with noticeable probability, given a function picked uniformly from H_λ. More precisely, we require that

$$\mathbf{Adv}^{\mathsf{cr}}_{\mathsf{H},\mathcal{A}}(\lambda) := \Pr[\mathrm{CR}^{\mathcal{A}}_{\mathsf{H}}(1^\lambda)] \in \textsc{Negl},$$

where game $\mathrm{CR}^{\mathcal{A}}_{\mathsf{H}}(1^\lambda)$ is defined in Figure 2.8.

```
$\underline{\mathrm{CR}^{\mathcal{A}}_{\mathsf{H}}(1^\lambda)}$:
$h \leftarrow_\$ \mathsf{H}_\lambda$
$(\mathsf{x}_0, \mathsf{x}_1) \leftarrow_\$ \mathcal{A}(1^\lambda, h)$
return $(\mathsf{x}_0 \neq \mathsf{x}_1 \wedge h(\mathsf{x}_0) = h(\mathsf{x}_1))$
```

Figure 2.8: Game defining collision-resistance of a hash function family H.

2.3.6 Puncturable pseudorandom functions

We define a puncturable pseudorandom function family $\mathsf{PPRF} := (\mathsf{PPRF.Gen}, \mathsf{PPRF.Eval}, \mathsf{PPRF.Punc})$ as the following triple of ppt algorithms. (1) $\mathsf{PPRF.Gen}$ on input the security parameter 1^λ outputs a uniform element in KSp_λ; (2) $\mathsf{PPRF.Eval}$ is deterministic and on input a key $\mathsf{k} \in \mathsf{KSp}_\lambda$ and a point $\mathsf{x} \in \mathsf{X}_\lambda$ outputs a point $\mathsf{y} \in \mathsf{Y}_\lambda$; (3) $\mathsf{PPRF.Punc}$ is probabilistic and on input a $\mathsf{k} \in \mathsf{KSp}_\lambda$ and a polynomial-size set of points $\mathsf{S} \subseteq \mathsf{X}_\lambda$ outputs a punctured key $\mathsf{k_S}$. As per [SW14], we require the PPRF to satisfy the following two properties:

Functionality preservation under puncturing : It holds that for every $\lambda \in \mathbb{N}$, every polynomial-size set $\mathsf{S} \subseteq \mathsf{X}_\lambda$ and every $\mathsf{x} \in \mathsf{X}_\lambda \setminus \mathsf{S}$,

$$\Pr\left[\mathsf{PPRF.Eval}(\mathsf{k},\mathsf{x}) = \mathsf{PPRF.Eval}(\mathsf{k_S},\mathsf{x}) \;\middle|\; \begin{array}{l} \mathsf{k} \leftarrow_\$ \mathsf{PPRF.Gen}(1^\lambda) \\ \mathsf{k_S} \leftarrow_\$ \mathsf{PPRF.Punc}(\mathsf{k},\mathsf{S}) \end{array} \right] = 1.$$

Pseudorandomness at punctured points : For every ppt adversary $\mathcal{A}$,

$$\mathbf{Adv}^{\mathsf{pprf}}_{\mathsf{PPRF},\mathcal{A}}(\lambda) := 2 \cdot \Pr[\mathrm{PPRF}^{\mathcal{A}}_{\mathsf{PPRF}}(1^\lambda)] - 1 \in \textsc{Negl},$$

where game $\mathrm{PPRF}^{\mathcal{A}}_{\mathsf{PPRF}}(1^\lambda)$ is defined in Figure 2.9.

```
$\underline{\mathrm{PPRF}^{\mathcal{A}}_{\mathsf{PPRF}}(1^\lambda)}$:
$(\mathsf{S}, st) \leftarrow_\$ \mathcal{A}_0(1^\lambda)$
$\mathsf{k} \leftarrow_\$ \mathsf{PPRF.Gen}(1^\lambda)$
$\mathsf{k_S} \leftarrow_\$ \mathsf{PPRF.Punc}(\mathsf{k}, \mathsf{S})$
$b \leftarrow_\$ \{0, 1\}$
$b' \leftarrow_\$ \mathcal{A}_1^{\textsc{Fn}}(1^\lambda, \mathsf{k_S}, st)$
return $(b = b')$

$\underline{\textsc{Fn}(\mathsf{x})}$:
if $\mathsf{x} \notin \mathsf{S}$ return $\mathsf{PPRF.Eval}(\mathsf{k_S}, \mathsf{x})$
if $T[\mathsf{x}] = \perp$ then
    $T[\mathsf{x}] \leftarrow_\$ \mathsf{Y}_\lambda$
if $b = 1$ return $T[\mathsf{x}]$
else return $\mathsf{PPRF.Eval}(\mathsf{k}, \mathsf{x})$
```

Figure 2.9: Game defining the pseudorandomness at punctured points property of a puncturable pseudorandom function PPRF.

Chapter 3

Unpredictable Samplers

The results described in this section have been published in [ABF16].

The privacy notions that we will be developing in the coming sections rely on multistage adversaries that must adhere to certain high-entropy requirements on the sampled circuits. Rather than speaking about specific distributions for specific circuit classes, we introduce a uniform treatment for *any* circuit class via an unpredictability game. Our framework allows one to introduce restricted classes of samplers by imposing structural restrictions on their internal operation *without changes* to the reference unpredictability game. Our framework extends that of Bellare, Stepanov and Tessaro [BST14] for obfuscators and also models the challenge-generation phase in private functional encryption in prior works [BRS13a, BRS13b, ATR14, AAB+15].

3.1 Definitions

Syntax. A sampler for a circuit family CSp is an algorithm $\mathcal{S}$ that on input the security parameter 1^λ and possibly some state information st outputs a pair of vectors of CSp_λ circuits $(\mathbf{C}_0, \mathbf{C}_1)$ of equal dimension, a pair of vectors of MSp_λ messages $(\mathbf{m}_0, \mathbf{m}_1)$ of equal dimension, and some auxiliary information z. We require the components of the two circuit (resp., message) vectors to be encoded as bit strings of equal length. Input st may encode information about the environment where the sampler is run (e.g., the public parameters of a higher-level protocol) and z models side information available on the sampled circuits or messages.

In the security games we will be considering later on, the goal of adversary will be to distinguish which of two circuit distributions produced by an unpredictable sampler was used to form some cryptographic data (e.g., an obfuscated circuit or an FE token). Our unpredictability definition formalizes the intuition that by examining the input/output behavior of the sampled circuits on messages of choice, the evaluation of legitimate circuits of choice on sampled messages, and the evaluation of sampled circuits on sampled messages, a point leading to differing outputs on some pair of sampled circuits cannot be found. Drawing a parallel to the functional encryption setting, once decryption tokens or encrypted messages become available, the tokens can be used by a legitimate adversary to compute the circuits underneath on arbitrary values, including some special messages that are possibly correlated with the circuits.

Unpredictability. A *legitimate* sampler $\mathcal{S}$ is *statistically (multi-instance) unpredictable* if for any unbounded *legitimate* predictor $\mathcal{P}$ that places polynomially many queries

$$\mathbf{Adv}^{\text{(m)pred}}_{\mathcal{S},\mathcal{P}}(\lambda) := \Pr\left[\text{(m)Pred}^{\mathcal{P}}_{\mathcal{S}}(1^\lambda)\right] \in \text{Negl} ,$$

where games $\text{mPred}^{\mathcal{P}}_{\mathcal{S}}(1^\lambda)$ and $\text{Pred}^{\mathcal{P}}_{\mathcal{S}}(1^\lambda)$ are shown in Figure 3.1. Sampler $\mathcal{S}$ is called legitimate if $\mathbf{C}_0(\mathbf{m}'_0) = \mathbf{C}_1(\mathbf{m}'_1)$ for all queries made to the Sp oracle in game $\text{mPred}^{\mathcal{P}}_{\mathcal{S}}(1^\lambda)$, or simply required that $\mathbf{C}_0(\mathbf{m}_0) = \mathbf{C}_1(\mathbf{m}_1)$ in game $\text{Pred}^{\mathcal{P}}_{\mathcal{S}}(1^\lambda)$. Predictor $\mathcal{P}$ is legitimate if $\mathsf{C}(\mathbf{m}_0) = \mathsf{C}(\mathbf{m}_1)$ for all queries made to the Func oracle (in both multi-instance and single-instance games).[1]

The mPred game is multi-instance and the predictor can place polynomially many queries to Sam and set st arbitrarily. The latter essentially ensures that $\mathcal{S}$ generates fresh entropy on any input st. We emphasize that the winning condition demands *component-wise* inequality of circuit outputs. In particular the predictor is *not* considered successful if it outputs a message which leads to different outputs across different Sam queries or within the same Sam query but on different circuit indices.

[1]We *do not* impose that $\mathbf{C}_0(\mathsf{m}) = \mathbf{C}_1(\mathsf{m})$ within the Func oracle as this is exactly the event that $\mathcal{P}$ is aiming to invoke to win the game. The restriction we do impose allows for a sampler to be unpredictable while possibility outputting low-entropy messages that might even differ on left and right.

```
$\underline{\mathrm{mPred}^{\mathcal{P}}_{\mathcal{S}}(1^\lambda)}$:
$(i, \mathsf{m}) \leftarrow_\$ \mathcal{P}^{\textsc{Sam},\textsc{Func},\textsc{Sp}}(1^\lambda)$
$(\mathbf{C}_0, \mathbf{C}_1, \mathbf{m}_0, \mathbf{m}_1) \leftarrow \mathsf{list}[i]$
return $(\mathbf{C}_0(\mathsf{m}) \neq \mathbf{C}_1(\mathsf{m}))$

$\underline{\textsc{Sam}(st)}$:
$(\mathbf{C}_0, \mathbf{C}_1, \mathbf{m}_0, \mathbf{m}_1, z) \leftarrow_\$ \mathcal{S}(st)$
$\mathsf{list} \leftarrow \mathsf{list} : (\mathbf{C}_0, \mathbf{C}_1, \mathbf{m}_0, \mathbf{m}_1)$
return $z$
```

```
$\underline{\textsc{Func}(i, \mathsf{m}, \mathsf{C})}$:
$(\mathbf{C}_0, \mathbf{C}_1, \mathbf{m}_0, \mathbf{m}_1) \leftarrow \mathsf{list}[i]$
return $(\mathbf{C}_0(\mathsf{m}), \mathsf{C}(\mathbf{m}_0))$

$\underline{\textsc{Sp}(i, j)}$:
$(\mathbf{C}_0, \mathbf{C}_1, \mathbf{m}_0, \mathbf{m}_1) \leftarrow \mathsf{list}[i]$
$(\mathbf{C}'_0, \mathbf{C}'_1, \mathbf{m}'_0, \mathbf{m}'_1) \leftarrow \mathsf{list}[j]$
return $\mathbf{C}_0(\mathbf{m}'_0)$
```

```
$\underline{\mathrm{Pred}^{\mathcal{P}}_{\mathcal{S}}(1^\lambda)}$:
$(st, st') \leftarrow_\$ \mathcal{P}_1(1^\lambda)$
$(\mathbf{C}_0, \mathbf{C}_1, \mathbf{m}_0, \mathbf{m}_1, z) \leftarrow_\$ \mathcal{S}(st)$
$\mathsf{m} \leftarrow_\$ \mathcal{P}_2^{\textsc{Func}}(1^\lambda, \mathbf{C}_0(\mathbf{m}_0), z, st')$
return $(\mathbf{C}_0(\mathsf{m}) \neq \mathbf{C}_1(\mathsf{m}))$

$\underline{\textsc{Func}(\mathsf{m}, \mathsf{C})}$:
return $(\mathbf{C}_0(\mathsf{m}), \mathsf{C}(\mathbf{m}_0))$
```

Figure 3.1: Games defining multi-instance and single-instance unpredictability of a sampler $\mathcal{S}$.

A number of technical choices have been made in devising these definitions. By the legitimacy of the sampler $\mathbf{C}_0(\mathbf{m}_0) = \mathbf{C}_1(\mathbf{m}_1)$ and hence only one of these values is provided to the predictor. Furthermore, since the goal of the predictor is to find a differing input, modifying the experiment so that Func returns $\mathbf{C}_1(\mathsf{m})$ (or both values) would result in an equivalent definition. Our definition intentionally does *not* consider unpredictability of messages. Instead, one could ask the predictor to output either a message that results in differing evaluations on challenge circuits or a circuit that evaluates differently on challenge messages. This would, however, lead to an excessively restrictive unpredictability notion and excludes many circuit samplers of practical relevance.

Composition. A standard guessing argument shows that any stateless sampler (one that keeps no internal state and uses independent random coins on each invocation, but might still receive st explicitly passed as input) is multi-instance unpredictable (where $\mathcal{P}$ can place q queries to Sam) if and only if it is single-instance unpredictable (where $\mathcal{P}$ can only place a single Sam query). The reduction in one direction is trivial. In the other direction we guess the index i^* that the multi-instance predictor $\mathcal{P}$ will output and simulate Sam queries $1, \dots, (i^* - 1)$ and $(i^* + 1), \dots, q$ by running the sampler $\mathcal{S}$ in the reduction—this is where we need the stateless property—and answer the i^*th one using the Sam oracle in the single-instance game. Queries to Func with index i^* are answered analogously using the single-instance counterpart whereas those with index different from i^* will use the explicit knowledge of the circuits and messages generated by the reduction.

Queries Sp with index (i, j) are answered as follows. If both i and j are different from i^*, use the explicit knowledge of circuits and messages. If $i = i^*$ but $j \neq i^*$, use the explicit knowledge of the messages and the single-instance Func oracle on i^*. If $i \neq i^*$ but $j = i^*$, use the knowledge of the circuit and single-instance Func. For (i^*, i^*) queries, use the Sp oracle in the single-instance game. Note that the legitimacy of the constructed single-instance predictor follows from the legitimacy of the multi-instance predictor *and* the sampler.

Proposition 1 (Unpredictability composition)**.** *A stateless sampler is multi-instance unpredictable (Figure 3.1 on the left) if and only if it is single-instance unpredictable (Figure 3.1 on the right).*

The samplers that we study in this work are stateless and therefore we use the definition in Figure 3.1 on the right for simplicity. Nevertheless, our framework can be used to analyze *stateful* samplers as well. We leave the study of such samplers as an important (and practically relevant) direction for future work. In the following section, we define a number of special classes of samplers by imposing structural restrictions on their internal operation. This serves to illustrate how various samplers that previously appeared in the literature can be modeled within our framework. In particular, definitions of high-entropy and block source samplers for keywords [BRS13a], block sources for inner products [BRS13b], and circuit sampler distributions used in various obfuscation definitions can be seen as particular cases within this framework.

3.2 Taxonomy of samplers

We define a number of special classes of samplers by imposing structural restrictions on their internal operation.

Stateless. The sampler does not keep any internal state and uses independent set of coins on each invocation. All samplers will be stateless in this work unless stated otherwise.

(t, s)**-bounded.** For polynomials t and s, with overwhelming probability $|\mathbf{C}_0| = |\mathbf{C}_1| \leq t(\lambda)$ and $|\mathbf{m}_0| = |\mathbf{m}_1| \leq s(\lambda)$.

Circuits-only. The sampler outputs no messages with overwhelming probability, i.e. it is $(\cdot, 0)$-bounded.

One-sided. $\mathbf{C}_0 = \mathbf{C}_1$ and $\mathbf{m}_0 = \mathbf{m}_1$ with overwhelming probability. In this case we will simply write $(\mathbf{C}, \mathbf{m}, z) \leftarrow_\$ \mathcal{S}(1^\lambda, st)$ for the sampling operation. Note that every one-sided sampler is trivially unpredictable.

Input-independent. For any 1^λ and st, $\mathcal{S}(1^\lambda, st) = \mathcal{S}(1^\lambda, \varepsilon)$ with overwhelming probability.

Aux-free. With overwhelming probability $z = \varepsilon$.

Simple. If the sampler is both aux-free and input-independent.

Random-aux. For a polynomial poly and a ppt algorithm $\mathcal{S}'$ the sampler takes the form

$$\begin{aligned} \mathcal{S}(1^\lambda, st): \quad & z \leftarrow_\$ \{0,1\}^{\mathsf{poly}(\lambda)}; \\ & (\mathbf{C}_0, \mathbf{C}_1, \mathbf{m}_0, \mathbf{m}_1) \leftarrow_\$ \mathcal{S}'(1^\lambda, z, st); \\ & \text{return } (\mathbf{C}_0, \mathbf{C}_1, \mathbf{m}_0, \mathbf{m}_1, z)\ . \end{aligned}$$

Differing-inputs. With overwhelming probability z contains the sampler's output circuits $(\mathbf{C}_0, \mathbf{C}_1)$. Note that statistical unpredictability would imply that the sampled circuits are *functionally equivalent*, whereas computational unpredictability would lead to a notion of differing-inputs samplers used to formulate differing-inputs obfuscation [ABG+13, BST14].

Block-source. A t-block-source is a random variable $X = (X_1, \ldots, X_t)$ where for every $j \in [t]$ and $x_1, \ldots, x_{j-1}$ it holds that $X_j|_{X_1=x_1,\ldots,X_{j-1}=x_{j-1}}$ has high min-entropy. There is therefore *sufficient* decorrelation between different components in such a distribution. We can model block sources in our framework by restricting attention to ppt samplers that take the form

$$\begin{aligned} \mathcal{S}(1^\lambda, st): \quad & (\mathbf{C}_0, \mathbf{C}_1) \leftarrow_\$ \mathcal{S}'(1^\lambda, st); \\ & j \leftarrow_\$ [t]; \\ & \text{return } ((\mathbf{C}_0[j], \mathbf{C}_1[j]), (\mathbf{C}_0[1..(j-1)], \mathbf{C}_1[1..(j-1)])) \end{aligned}$$

where $\mathcal{S}'$ is a $(t, 0)$-bounded sampler. The rationale here is that any indistinguishability-based security definition that imposes an adversary to output two block sources, and later on distinguish some computation performed on the sampled values, e.g. [BRS13a], would remain the same if a sampler such as the one above was used instead (note that in this case, the adversary can only have an advantage when outputting distributions that component-wise differ with non-negligible probability).

Functionally-differing. $\mathcal{S}$ is a $(t, \cdot)$-bounded sampler and, with overwhelming probability over its random coins, $\forall j \in [t]$, $\exists x$ s.t. $\mathbf{C}_0[j](x) \neq \mathbf{C}_1[j](x)$.

Chapter 4

Obfuscators

The results described in this section have been published in [ABF16].

An obfuscator is an efficient compiler that takes as input a circuit description and outputs the description of another circuit with the same functionality that is of similar size, but yet is *unintelligible*. In recent years, program obfuscation as found tremendous applications in cryptography, ranging from functional encryption [GGH+13] to deniable encryption [SW14] to multi-party computation [GGHR14]. Several of the constructions we propose later in this thesis rely on obfuscation, so we take the time go over definitions and revisit concrete obfuscators for the classes of functions comprising point functions and hyperplane membership testing functions. We generalize distributionally-indistinguishable (DI) obfuscators [BC14] from point functions to general circuits, show that DI is a weakening of composable virtual grey-box (CVGB) [BC14, BCKP14], and construct a DI obfuscator for hyperplane membership testing functions.

Syntax. An obfuscator for a circuit family CSp is a uniform ppt algorithm Obf that on input the security parameter 1^λ and the description of a circuit $\mathsf{C} \in \mathsf{CSp}_\lambda$ outputs the description of another circuit $\overline{\mathsf{C}}$. We require any obfuscator to satisfy the following two requirements.

Functionality preservation : For any $\lambda \in \mathbb{N}$, any $\mathsf{C} \in \mathsf{CSp}_\lambda$ and any $\mathsf{m} \in \mathsf{MSp}_\lambda$, with overwhelming probability over the choice of $\overline{\mathsf{C}} \leftarrow_{\$} \mathsf{Obf}(1^\lambda, \mathsf{C})$ we have that $\mathsf{C}(\mathsf{m}) = \overline{\mathsf{C}}(\mathsf{m})$.

Polynomial slowdown : There is a polynomial poly such that for any $\lambda \in \mathbb{N}$, any $\mathsf{C} \in \mathsf{CSp}_\lambda$ and any $\overline{\mathsf{C}} \leftarrow_\$ \mathsf{Obf}(1^\lambda, \mathsf{C})$ we have that $|\overline{\mathsf{C}}| \leq \mathsf{poly}(|\mathsf{C}|)$.

Security definitions for obfuscators can be divided into indistinguishability-based and simulation-based notions. Perhaps the most natural notion is the virtual black-box (VBB) property [BGI+01], which requires that whatever can be computed from an obfuscated circuit can be also simulated using oracle access to the circuit. Here, we consider a weakening of this notion, known as *virtual grey-box* (VGB) security [BC14, BCKP14] that follows the VBB approach, but allows simulators to run in *unbounded* time, as long as they make polynomially many queries to their oracles; we call such simulators semi-bounded. Below we present a self-composable strengthening of this notion where the VGB property is required to hold in the presence of multiple obfuscated circuits.

In the context of security definitions for obfuscators, we consider samplers that do not output any messages. Furthermore, we call a sampler *one-sided* if its sampled circuits are identical on left and right with probability 1.

4.1 Indistinguishability obfuscation

This property requires that given any two functionally equivalent circuits C_0 and C_1 of equal size, the obfuscations of C_0 and C_1 should be computationally indistinguishable. More precisely, for any ppt adversary $\mathcal{A}$ and for any sampler $\mathcal{S}$ that outputs two circuits $\mathsf{C}_0, \mathsf{C}_1 \in \mathsf{CSp}_\lambda$ such that $\mathsf{C}_0(\mathsf{m}) = \mathsf{C}_1(\mathsf{m})$ for all inputs m and $|\mathsf{C}_0| = |\mathsf{C}_1|$, we have that

$$\mathbf{Adv}^{\mathrm{io}}_{\mathsf{Obf},\mathcal{S},\mathcal{A}}(\lambda) := 2 \cdot \Pr[\mathrm{iO}^{\mathcal{S},\mathcal{A}}_{\mathsf{Obf}}(1^\lambda)] - 1 \in \textsc{Negl},$$

where game $\mathrm{iO}^{\mathcal{S},\mathcal{A}}_{\mathsf{Obf}}(\lambda)$ is defined in Figure 4.1.

```
$\underline{\mathrm{iO}^{\mathcal{S},\mathcal{A}}_{\mathsf{Obf}}(1^\lambda)}$:
$(\mathsf{C}_0, \mathsf{C}_1, z) \leftarrow_\$ \mathcal{S}(1^\lambda)$
$b \leftarrow_\$ \{0,1\}$
$\overline{\mathsf{C}} \leftarrow_\$ \mathsf{Obf}(1^\lambda, \mathsf{C}_b)$
$b' \leftarrow_\$ \mathcal{A}_1(1^\lambda, \mathsf{C}_0, \mathsf{C}_1, z, \overline{\mathsf{C}})$
return $(b = b')$
```

Figure 4.1: Game defining iO security of an obfuscator Obf.

4.2 Composable VGB obfuscation

An obfuscator Obf is composable VGB (CVGB) secure if for every ppt adversary $\mathcal{A}$ there exists a semi-bounded simulator Sim such that for every ppt one-sided circuit sampler $\mathcal{S}$ the advantage

$$\mathbf{Adv}^{\mathrm{cvgb}}_{\mathsf{Obf},\mathcal{S},\mathcal{A},\mathsf{Sim}}(\lambda) := \left|\Pr\left[\mathrm{CVGB\text{-}Real}^{\mathcal{S},\mathcal{A}}_{\mathsf{Obf}}(1^\lambda)\right] - \Pr\left[\mathrm{CVGB\text{-}Ideal}^{\mathcal{S},\mathsf{Sim}}_{\mathsf{Obf}}(1^\lambda)\right]\right| \in \textsc{Negl},$$

where games $\mathrm{CVGB\text{-}Real}^{\mathcal{S},\mathcal{A}}_{\mathsf{Obf}}(\lambda)$ and $\mathrm{CVGB\text{-}Ideal}^{\mathcal{S},\mathsf{Sim}}_{\mathsf{Obf}}(\lambda)$ are shown in Figure 4.2.

```
$\underline{\mathrm{CVGB\text{-}Real}^{\mathcal{S},\mathcal{A}}_{\mathsf{Obf}}(1^\lambda)}$:
$(\mathbf{C}, z) \leftarrow_\$ \mathcal{S}(1^\lambda, \varepsilon)$
$\overline{\mathbf{C}} \leftarrow_\$ \mathsf{Obf}(1^\lambda, \mathbf{C})$
$b \leftarrow_\$ \mathcal{A}(1^\lambda, \overline{\mathbf{C}}, z)$
return $b$

$\underline{\mathrm{CVGB\text{-}Ideal}^{\mathcal{S},\mathsf{Sim}}_{\mathsf{Obf}}(1^\lambda)}$:
$(\mathbf{C}, z) \leftarrow_\$ \mathcal{S}(1^\lambda, \varepsilon)$
$b \leftarrow_\$ \mathsf{Sim}^{\textsc{Func}}(1^\lambda, 1^{|\mathbf{C}|}, z)$
return $b$

$\underline{\textsc{Func}(\mathsf{m})}$:
return $\mathbf{C}(\mathsf{m})$
```

Figure 4.2: Games defining the CVGB security of an obfuscator Obf.

By considering samplers that only output a single circuit we recover the standard (worst-case) VGB property. The VBB property corresponds to the case where the simulator is required to run in polynomial time. Average-case notions of obfuscation correspond to definitions where the circuit samplers are fixed. A result of Bitansky and Canetti [BC14, Proposition A.3] on the equivalence of VGB *with* and *without* auxiliary information can be easily shown to also hold in the presence of multiple circuits, from which one can conclude that CVGB *with* auxiliary information is the same as CVGB *without* auxiliary information.

We also introduce the following adaptation of an indistinguishability-based notion of obfuscation introduced in [BC14] for point functions.

4.3 Distributional indistinguishability obfuscation

An obfuscator Obf is DI secure if, for every unpredictable ppt sampler $\mathcal{S}$ and every ppt adversary $\mathcal{A}$,

$$\mathbf{Adv}^{\mathrm{di}}_{\mathsf{Obf},\mathcal{S},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathrm{DI}^{\mathcal{S},\mathcal{A}}_{\mathsf{Obf}}(1^\lambda)\right] - 1 \in \textsc{Negl} \ ,$$

where game $\mathrm{DI}^{\mathcal{S},\mathcal{A}}_{\mathsf{Obf}}(1^\lambda)$ is defined in Figure 4.3 on the left.

```
$\underline{\mathrm{DI}^{\mathcal{S},\mathcal{A}}_{\mathsf{Obf}}(1^\lambda)}$:
$b \leftarrow_\$ \{0,1\}$
$b' \leftarrow_\$ \mathcal{A}^{\textsc{Sam}}(1^\lambda)$
return $(b = b')$

$\underline{\textsc{Sam}(st)}$:
$(\mathbf{C}_0, \mathbf{C}_1, z) \leftarrow_\$ \mathcal{S}(1^\lambda, st)$
$\overline{\mathbf{C}} \leftarrow_\$ \mathsf{Obf}(1^\lambda, \mathbf{C}_b)$
return $(\overline{\mathbf{C}}, z)$
```

```
$\underline{\mathrm{OW}^{\mathcal{A}}_{\mathsf{Obf}}(1^\lambda)}$:
$1^t \leftarrow_\$ \mathcal{A}_1(1^\lambda)$
$\mathsf{w} \leftarrow_\$ \mathsf{MSp}_\lambda$
$\mathbf{C} \leftarrow [\mathsf{C}[\mathsf{w}], \ldots, \mathsf{C}[\mathsf{w}]]$ // $t$ copies
$\overline{\mathbf{C}} \leftarrow_\$ \mathsf{Obf}(1^\lambda, \mathbf{C})$
$\mathsf{w}' \leftarrow_\$ \mathcal{A}_2(1^\lambda, \overline{\mathbf{C}})$
return $(\mathsf{w} = \mathsf{w}')$
```

Figure 4.3: Games defining the DI and OW security of an obfuscator Obf. (One-way security is defined for point-functions only.)

The above definition strengthens the one in [BC14] and gives the sampler the possibility to leak auxiliary information to the adversary. In particular, we can consider the case where images of an (internally generated) vector of messages that are correlated with the circuits are provided to $\mathcal{A}$. (Our constructions will rely on this property for point obfuscators.) Throughout the thesis we consider DI adversaries that place a single query to the Sam oracle. It can easily be shown that the DI self-composes for stateless samplers, meaning that security against adversaries that place one Sam query is equivalent to the setting where an arbitrary number of queries are allowed. Note also that we allow the adversary to pass some state information st to the sampler. Security with respect to *all* ppt and statistically unpredictable samplers can be shown to be equivalent to a variant definition where the adversary is run *after* the sampler and the state st is set to the empty string ε.

We recover the definition of indistinguishability obfuscation (iO) [GGH+13] when samplers are required to output a single circuit on left and right and include these two circuits explicitly in z. Differing-inputs obfuscation (diO) [ABG+13] is obtained if the predictor is also limited to run in polynomial time.

It has been shown that, for *point functions*, the notions of CVGB and DI (without auxiliary information) are equivalent [BC14, Theorem 5.1]. Following a similar argument to the first part of the proof in [BC14, Theorem 5.1], we show that CVGB for *any* circuit family implies distributional indistinguishability even *with auxiliary information* for the same circuit family. Hence, our notion of DI obfuscation is potentially *weaker* than CVGB. This proof crucially relies on the restriction that samplers are required to be unpredictable in the presence of unbounded predictors. The proof of the converse direction in [BC14, Theorem 5.1] uses techniques specific to point functions and we leave a generalization to wider classes of circuits for future work.

Proposition 2 (CVGB $\implies$ DI)**.** *Any* CVGB *obfuscator for a class of circuits* CSp *is also* DI *secure with respect to all statistically unpredictable samplers for the same class* CSp.

Proof. Let $(\mathcal{S}, \mathcal{A})$ be a DI adversary against the obfuscator. We show that the advantage of $\mathcal{A}$ must be negligible if $\mathcal{S}$ is unpredictable and the obfuscator is CVGB secure. Also, let RSp_λ denote the randomness space of $\mathcal{A}$. Consider a one-sided circuit sampler $\mathcal{S}'$ that selects $\mathsf{r} \leftarrow_\$ \mathsf{RSp}_\lambda$, runs $\mathcal{A}(1^\lambda; \mathsf{r})$ until it outputs st, runs $\mathcal{S}(1^\lambda, st)$, chooses a bit b uniformly at random, and outputs the left or right outputs of $\mathcal{S}$ according to the bit b, along with auxiliary information z and coins r. Let $\mathcal{B}$ be a CVGB-Real adversary that runs $\mathcal{A}$ on the same coins and answers SAM oracle query with its challenge vector of obuscations. $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. By the CVGB property, for $(\mathcal{S}', \mathcal{B})$ there is a (possibly unbounded) simulator Sim such that:

$$\mathbf{Adv}^{\mathrm{cvgb}}_{\mathsf{Obf},\mathcal{S}',\mathcal{B},\mathsf{Sim}}(\lambda) = \left| \Pr\left[\text{CVGB-Real}^{\mathcal{S}',\mathcal{B}}_{\mathsf{Obf}}(1^\lambda)\right] - \Pr\left[\text{CVGB-Ideal}^{\mathcal{S}',\mathsf{Sim}}_{\mathsf{Obf}}(1^\lambda)\right] \right| .$$

Note that

$$\mathbf{Adv}^{\mathrm{di}}_{\mathsf{Obf},\mathcal{S},\mathcal{A}}(\lambda) = \Pr\left[\text{CVGB-Real}^{\mathcal{S}',\mathcal{B}}_{\mathsf{Obf}}(1^\lambda)|b=1\right] - \Pr\left[\text{CVGB-Real}^{\mathcal{S}',\mathcal{B}}_{\mathsf{Obf}}(1^\lambda)|b=0\right] .$$

Hence,

$$\mathbf{Adv}^{\mathrm{di}}_{\mathsf{Obf},\mathcal{S},\mathcal{A}}(\lambda) \leq \left|\Pr\left[\text{CVGB-Ideal}^{\mathcal{S}',\mathsf{Sim}}_{\mathsf{Obf}}(1^\lambda)|b=1\right] - \Pr\left[\text{CVGB-Ideal}^{\mathcal{S}',\mathsf{Sim}}_{\mathsf{Obf}}(1^\lambda)|b=0\right]\right| + \\ + 2 \cdot \mathbf{Adv}^{\mathrm{cvgb}}_{\mathsf{Obf},\mathcal{S}',\mathcal{B},\mathsf{Sim}}(\lambda) \ .$$

Let $Q(\lambda)$ denote the number of queries of Sim. We claim that there is a predictor $\mathcal{P}$ making at most $Q(\lambda)$ queries such that

$$\left|\Pr\left[\text{CVGB-Ideal}^{\mathcal{S}',\mathsf{Sim}}_{\mathsf{Obf}}(1^\lambda)|b=1\right] - \Pr\left[\text{CVGB-Ideal}^{\mathcal{S}',\mathsf{Sim}}_{\mathsf{Obf}}(1^\lambda)|b=0\right]\right| \leq Q(\lambda)\cdot\mathbf{Adv}^{\mathrm{pred}}_{\mathcal{S},\mathcal{P}}(\lambda) \ .$$

From this it follows that

$$\mathbf{Adv}^{\mathrm{di}}_{\mathsf{Obf},\mathcal{S},\mathcal{A}}(\lambda) \leq Q(\lambda)\cdot\mathbf{Adv}^{\mathrm{pred}}_{\mathcal{S},\mathcal{P}}(\lambda) + 2\cdot\mathbf{Adv}^{\mathrm{cvgb}}_{\mathsf{Obf},\mathcal{S}',\mathcal{B},\mathsf{Sim}}(\lambda) \ .$$

We prove the claim via unpredictability of the sampler. Observe that the views of Sim in the CVGB-Ideal game for $b = 0$ and $b = 1$ are identical unless Sim queries its oracle on a point that results in different outputs for the left and right circuits. This event, however, immediately leads to a break of unpredictability. Consider a (possibly unbounded) predictor $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$ as follows. $\mathcal{P}_1$ selects random coins $\mathsf{r} \leftarrow_{\$} \mathsf{RSp}_\lambda$ and runs $\mathcal{A}(1^\lambda;\mathsf{r})$ until it outputs st. $\mathcal{P}_1$ then outputs (st,r). $\mathcal{P}_2(1^\lambda,\epsilon,z,\mathsf{r})$ chooses a random index $i \leftarrow_{\$} [Q(\lambda)]$ indicating a guess for the first query of Sim that leads to a break of unpredictability. It runs $\mathsf{Sim}(z||\mathsf{r})$ and answers its oracle queries using its own provided oracle (which always respond for left circuits $b = 0$). At query i algorithm $\mathcal{P}_2$ stops and outputs the queried value. With probability $1/Q(\lambda)$ this is the first query that the bad event occurs. Hence $\mathcal{P}_2$ runs Sim perfectly until query i, at which point it wins the unpredictability game.

This concludes the proof as the above holds for any poly (which in turn implies that the left hand side is negligible). □

We conclude this discussion by introducing a new notion of *one-way* point obfuscation that requires it to be infeasible to recover the point given many obfuscations of it.

ONE-WAY POINT OBFUSCATION. Let Obf be an obfuscator for a point circuit family CSp. We say Obf is OW secure if for every ppt adversary $\mathcal{A}$

$$\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{Obf},\mathcal{A}}(\lambda) := \Pr\left[\mathrm{OW}^{\mathcal{A}}_{\mathsf{Obf}}(1^\lambda)\right] \in \mathrm{NEGL} \ ,$$

where game $\mathrm{OW}^{\mathcal{A}}_{\mathsf{Obf}}(1^\lambda)$ is shown in Figure 4.3 on the right. The next proposition shows that OW is a weakening of DI for point circuits.

Proposition 3 (DI $\implies$ OW for point circuits)**.** *Let* Obf *be an obfuscator for a point circuit family* CSp*. If* Obf *is* DI *secure with respect to all ppt samplers, then it is also* OW *secure.*

Proof. We show that OW is a weakening of DI for point circuits. Given an OW adversary $\mathcal{A}$ we construct a sampler $\mathcal{S}$ and a distinguisher $\mathcal{D}$ attacking DI security as follows. First, we partition each set CSp_λ into two sets (of super-polynomial size) CSp^0_λ and CSp^1_λ, such that $|\mathsf{CSp}^0_\lambda| = |\mathsf{CSp}^1_\lambda| + \mathsf{negl}(\lambda)$. This partition can be based on some lexicographic criterion (e.g., the most significant bit of the point), as long as one can efficiently decide membership in each partition. Our sampler $\mathcal{S}$ samples two point circuits C_0 and C_1, uniformly at random from CSp^0_λ and CSp^1_λ, respectively. It then outputs two t-sized vectors $\mathbf{C}_0 = (\mathsf{C}_0, \ldots, \mathsf{C}_0)$ and $\mathbf{C}_1 = (\mathsf{C}_1, \ldots, \mathsf{C}_1)$. (Here t is the length parameter initially output by the one-wayness adversary $\mathcal{A}$.) (Recall that auxiliary information z is empty.) It is clear that $\mathcal{S}$ is unpredictable, and therefore legitimate as a DI sampler. On obtaining the obfuscations, the distinguisher $\mathcal{D}$ runs adversary $\mathcal{A}$ on the same inputs and recover a circuit C'. Observe that the distribution of the obfuscations provided to $\mathcal{A}$ is statistically close to the correct distribution given the *combined* action of $\mathcal{S}$ and the challenge bit in the DI game. Distinguisher $\mathcal{D}$ then returns 0 if $\mathsf{C}' \in \mathsf{CSp}^0_\lambda$ and 1 otherwise. It is straightforward to establish that a non-negligible advantage for $\mathcal{A}$ in the OW game translates to a non-negligible advantage for $(\mathcal{S}, \mathcal{D})$ in the DI game. □

4.3.1 Canetti's point obfuscator

In this section, we recall the point function obfuscator constructed in [Can97] and proven to be DI-secure under the Strong Vector DDH (SVDDH) assumption in [BC14]. Not only this point function obfuscator can be used later to instantiate our OX, TOX and DOX constructions (see Chapter 6), we use a similar approach to prove that the hyperplane membership obfuscator by [CRV10] is also DI-secure under a generalization of the SVDDH assumption. (We refer to the next section for more details on these results.)

Let G be a group of prime order p for which the SVDDH assumption [BC14] holds, and let $\mathsf{CSp} := \{\mathsf{C}[x] : \mathsf{m} \in \mathbb{Z}_p\}$ be a point circuit family for points in the domain $\mathbb{Z}_p$. To obfuscate the point circuit $\mathsf{C}[x]$, first sample a generator r uniformly at random from G, compute $R \leftarrow g^x$, then construct the circuit $\mathsf{C}[r, R]$, which has r and R hardwired into it, such that:

$$\mathsf{C}[r, R](z) := \begin{cases} 1 & \text{if } r^z = R; \\ 0 & \text{otherwise.} \end{cases}$$

Under the SVDDH assumption [BC14, Assumption 6.1], this obfuscator can be shown to be DI secure [BC14, Theorem 6.1].

4.3.2 An hyperplane membership obfuscator

Let $\mathsf{CSp} := \{\mathsf{CSp}_p^d\}$ be a set circuit family of hyperplane membership testing functions that is defined for each value of the security parameter λ such that there is a λ-bit prime p and a positive integer d. Every circuit $\mathsf{C} \in \mathsf{CSp}_p^d$ is canonically represented by a vector $\vec{a} \in \mathbb{Z}_p^d$ and returns 1 if and only if the input vector $\vec{x} \in \mathbb{Z}_p^d$ is *orthogonal* to $\vec{a}$, i.e.,

$$\mathsf{C}[\vec{a}](\vec{x}) := \begin{cases} 1 & \text{if } \langle \vec{x}, \vec{a} \rangle = 0; \\ 0 & \text{otherwise.} \end{cases}$$

We build on the results of [BC14, CRV10] to construct a DI-secure obfuscator for this family of circuits under a generalization of the Strong Vector DDH (SVDDH) assumption used in [BC14]. In order to avoid attacks similar to the one described in [BST16] that puts a one element instance of SVDDH with *arbitrary* auxiliary information (or AI-DHI assumption, as referred to by [BST16]) in contention with the existence of VGB obfuscators supporting specific classes of circuits, we assume that our generalized SVDDH assumption holds only in the presence of *random* auxiliary information. This immediately translates to an obfuscator that tolerates the same type of leakage, which is enough to serve as a candidate to instantiate our functionality-agnostic constructions and obtain private inner-product encryption schemes, from which it is known how to derive expressive predicates that include equality tests, conjunctions, disjunctions and evaluation of CNF and DNF formulas (among others) [BW07, KSW13].

Canetti, Rothblum and Varia [CRV10] presented a virtual black-box obfuscator for the hyperplane membership functionality, which works as follows. Let G be a group of prime order p for which the SVDDH assumption [CRV10] holds. To obfuscate the hyperplane membership circuit represented by a vector $\vec{a}$, sample a generator g uniformly at random from G, compute $g_i \leftarrow g^{\vec{a}[i]}$ for $1 \leq i \leq d$, and construct the circuit that, given a vector $\vec{x}$, returns 1 if and only if $\prod_{i=1}^{d} g_i^{\vec{x}[i]}$ is equal to G's identity element. (Note that $\prod_{i=1}^{d} g_i^{\vec{x}[i]} = g^{\langle \vec{a}, \vec{x} \rangle}$, so this is the case if $\langle \vec{a}, \vec{x} \rangle = 0$.) We assume that the resulting obfuscated circuit is canonically represented by $(g_1, \ldots, g_d)$, generated as described above. We will now prove that this same construction satisfies distributional indistinguishability under a generalization of the SVDDH assumption, a DDH-style assumption we present in Figure 4.5.

Unpredictable hyperplane membership samplers. We begin by refining the notion of unpredictable samplers to the case of hyperplane membership circuits. In general, a sampler for the hyperplane membership functionality will output two lists of message vectors corresponding to candidate hyperplane members, and two lists of hyperplane vectors, plus some auxiliary information z, which in this thesis we will assume to be a random string of polynomial size $\mathsf{poly}(\lambda)$. However, since we are dealing with obfuscation, we will consider samplers where no messages are produced. We recall the unpredictability experiment for this special case in Figure 4.4, where notation $\langle \mathbf{w}, \mathsf{m} \rangle$ denotes the vector that results from computing $\langle \mathbf{w}[i], \mathsf{m} \rangle \stackrel{?}{=} 0$, for all $1 \leq i \leq |\mathbf{w}|$. (Note that here $\mathbf{w}$ is a *list* of vectors in $\mathbb{Z}_p^d$ and m is a vector in $\mathbb{Z}_p^d$.) Sampler outputs only *random* auxiliary information.

```
$\underline{\mathrm{Pred}_{\mathcal{S}}^{\mathcal{P}}(1^\lambda)}$:
$(st, st') \leftarrow\$ \mathcal{P}_1(1^\lambda)$
$z \leftarrow\$ \{0,1\}^{\mathsf{poly}(\lambda)}$
$(\mathbf{w}_0, \mathbf{w}_1) \leftarrow\$ \mathcal{S}(1^\lambda, z, st)$
$\mathsf{m} \leftarrow\$ \mathcal{P}_2^{\mathrm{FUNC}}(1^\lambda, z, st')$
return $(\langle \mathbf{w}_0, \mathsf{m} \rangle \neq \langle \mathbf{w}_1, \mathsf{m} \rangle)$

$\underline{\mathrm{FUNC}(\mathsf{m})}$:
return $\langle \mathbf{w}_0, \mathsf{m} \rangle$
```

Figure 4.4: Game defining single-instance unpredictability of a hyperplane membership sampler $\mathcal{S}$.

Computational Assumption. Our computational assumption is a vectorized version of the DDH variant introduced in [CRV10], in the style of the assumption that is used in [BC14] to establish the DI property of a point function obfuscator. The assumption

```
$\underline{\mathsf{Assumption}_{\mathcal{S},\mathcal{A},G,t,d,\mathsf{poly}}(1^\lambda)}$:
$b \leftarrow_\$ \{0,1\}$; $z \leftarrow_\$ \{0,1\}^{\mathsf{poly}(\lambda)}$
$(\mathbf{w}_0, \mathbf{w}_1) \leftarrow_\$ \mathcal{S}(1^\lambda, z, \epsilon)$
$\vec{g} \leftarrow_\$ G^t$
$M_b \leftarrow \begin{bmatrix} \vec{g}[1]^{\mathbf{w}_b[1][1]} & \dots & \vec{g}[1]^{\mathbf{w}_b[1][d]} \\ \dots & \dots & \dots \\ \vec{g}[t]^{\mathbf{w}_b[t][1]} & \dots & \vec{g}[t]^{\mathbf{w}_b[t][d]} \end{bmatrix}$
$b' \leftarrow_\$ \mathcal{A}(M_b, z)$
return $(b = b')$
```

Figure 4.5: Game defining a DDH-style computational assumption.

states that, for every unpredictable sampler, any distinguishing adversary has a negligible advantage in the game in Figure 4.5. We note that the unpredictability restriction on the sampler essentially excludes any challenge where a polynomial-size set of black-box linear tests could be used by a semi-bounded predictor to distinguish the hidden bit. This is a natural restriction, since the adversary is given enough information to trivially perform such tests on its own. The assumption therefore states that ppt adversaries cannot do better than what can be achieved with such linear tests. In particular, we note that such linear tests can be used to extract coefficient equality patterns that might permit trivial distinguishing attacks by checking group element repetitions in the received obfuscations.

DI OBFUSCATION FOR HYPERPLANE MEMBERSHIP. The following theorem can be trivially proven using a direct reduction.

Theorem 1. *The hyperplane membership obfuscator of Canetti, Rothblum and Varia [CRV10] is DI secure in the presence of random auxiliary information if the assumption in Figure 4.5 holds in $\mathcal{G}$. More precisely, for every unpredictable hyperplane membership sampler $\mathcal{S}$, any DI adversary $\mathcal{A}$ that breaks the DI property can be used (without change) to break the underlying assumption with the same advantage.*

4.4 Instantiations and obfuscation-based attacks

A concrete instantiation of a CVGB obfuscator for point functions *with* auxiliary information (AI) is given by Bitansky and Canetti [BC14]. This construction is based on the hardness of a variant of the DDH assumption called strong vector-DDH (SVDDH) assumption. The SVDDH assumption is an assumption that is formulated *without* reference to any auxiliary information. Recently, Bellare, Stepanovs and Tessaro [BST16] have shown that the SVDDH assumption (and *verifiable* point obfuscation) in presence of arbitrary AI is in contention with the existence of VGB obfuscation for general circuits (that is, one of the two cannot exist). We take a moment to clarify how these two results relate to each other. In this discussion we assume that all obfuscation notions are considered for a single circuit only (i.e. we do not consider composability). First, note that the notion of AIPO (auxiliary-information point obfuscation) used in [BST16] follows a notion equivalent to distributional indistinguishability where the right distribution is fixed to be uniform. As shown in [BC14, Theorem 5.1] any point obfuscation (without AI) is equivalent to VGB point obfuscation. It is also shown in [BC14, Proposition A.3] that VGB obfuscation without AI is equivalent to VGB obfuscation *with* AI for any circuit class. (Intuitively, to construct a simulator that works for all possible AI, one uses the fact that the simulator is unbounded to find the best simulator that works for a non-uniform adversary that takes a value of the AI as advice.) Together with Proposition 2 above we get that all these notions (in their non-composable variants) are *equivalent*. This then raises the question whether the results of [BST16] are also in contention with PO without AI. To see that this does not follow from the equivalence of notions, note that in Proposition 2 we crucially rely on a predictor that runs a possibly unbounded simulator. Put differently, the AI must be *statistically* unpredictable. Indeed, the results of [BST16] rely on special forms of AI which only computationally hide the sampled point. (Roughly speaking, the AI contains a VGB obfuscation of a (non-point) circuit that depends on the sampled point.) To avoid such attacks, and in line with the above results, by considering statistically unpredictable samplers we constrain auxiliary information to be statistically unpredictable.

Chapter 5

Function Privacy: A Unified Approach

The results described in this section have been published in [ABF16].

In this chapter, we define what function privacy for general functional encryption schemes means and derive the model specific to keyword search schemes by restriction to point circuit families. Our definition follows the indistinguishability-based approach to defining FE security and comes with an analogous legitimacy condition that prevents the adversary from learning the challenge bit simply by extracting a token for a circuit that has differing outputs for the left and right challenge messages. The model extends the IND-CPA game via a left-or-right (LR) oracle that returns ciphertexts *and* tokens for possibly correlated messages and circuits. Since the adversary in this game has access to tokens that depend on the challenge bit, we use the unpredictability framework of Chapter 3 to rule out trivial guess attacks.

The game follows a left-or-right rather than a real-or-random formulation of the challenge oracle [BRS13a, BRS13b, ATR14, AAB^{+}15] as this choice frees the definition from restrictions that must be imposed to render samplers compatible with uniform distribution over circuits. In particular, it allows the sampler to output *low-entropy* circuits as long as they are functionally-equivalent on left and right. It also allows analyzing security under repetitions of functionally-equivalent circuits in the presence of correlated messages, which until now were properties captured separately by *unlinkability* [ATR14] and *enhanced security* [BRS13a], and never considered together, not even for the simple case of point functions.

The sampler allows us to model, within a single game, (a) token-only adversarial strategies via samplers that output no message, as the *non-enhanced* security model in [BRS13a] and those in [BRS13b, ATR14]; (b) adversarial strategies that admit simple correlations between encrypted messages and extracted circuits, as the *enhanced* security model in [BRS13a] for point circuits that allows the adversary to obtain ciphertexts that *match* the tokens; (c) adversarial strategies that admit *arbitrary* correlations between extracted circuits and encrypted messages (i.e., not only exact matches).

Our model is functionality-agnostic and unifies all previous indistinguishability-based models in this area. When restricted to point circuits or inner-products families, it gives rise to a new privacy notion that offers significant improvements over those in prior works [BRS13a, BRS13b, ATR14].

PRIV SECURITY. A functional encryption scheme FE is PRIV secure if, for every unpredictable ppt sampler[1] $\mathcal{S}$ and every ppt adversary $\mathcal{A}$

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathsf{FE},\mathcal{A},\mathcal{S}}(\lambda) := 2 \cdot \Pr\left[\mathrm{PRIV}^{\mathcal{A},\mathcal{S}}_{\mathsf{FE}}(1^\lambda)\right] - 1 \in \textsc{Negl} \ ,$$

where game $\mathrm{PRIV}^{\mathcal{A},\mathcal{S}}_{\mathsf{FE}}(1^\lambda)$ is defined in Figure 5.1. We exclude adversaries $(\mathcal{A},\mathcal{S})$ that attempt to trivially win the PRIV game via decryption tokens, by either extracting them explicitly via the token-generation oracle, or implicitly via the left-or-right oracle. Formally, the pair $(\mathcal{A},\mathcal{S})$ is legitimate if, with overwhelming probability

$$\forall (\mathbf{C}_0, \mathbf{C}_1) \in \mathsf{TList}\, , \forall (\mathbf{m}_0, \mathbf{m}_1) \in \mathsf{MList} : \mathbf{C}_0(\mathbf{m}_0) = \mathbf{C}_1(\mathbf{m}_1) \ .$$

Note also that for two sampler classes $\mathbb{S}_1$ and $\mathbb{S}_2$ with $\mathbb{S}_1 \subset \mathbb{S}_2$ security with respect to samplers in $\mathbb{S}_2$ is a stronger security guarantee that one for those only in $\mathbb{S}_1$. In particular a stronger restriction on sampler classes results in a weaker definition.

The definition also provides the adversary with the ability to adaptively obtain multiple challenges and tokens. However, similarly to unpredictability, a hybrid argument

[1]We limit samplers to ppt because in proving the security of our constructions, samplers are used to construct computational adversaries against other schemes. In general, one could consider unbounded samplers.

```
$\underline{\mathrm{PRIV}^{\mathcal{A},\mathcal{S}}_{\mathsf{FE}}(1^\lambda)}$:
    $(\mathsf{msk}, \mathsf{mpk}) \leftarrow_\$ \mathsf{FE.Gen}(1^\lambda)$
    $b \leftarrow_\$ \{0,1\}$
    $b' \leftarrow_\$ \mathcal{A}^{\mathrm{LR},\mathrm{TGen}}(\mathsf{mpk})$
    return $(b = b')$

$\underline{\mathrm{LR}(st)}$:
    $(\mathbf{C}_0, \mathbf{C}_1, \mathbf{m}_0, \mathbf{m}_1, z) \leftarrow_\$ \mathcal{S}(st)$
    $\mathsf{TList} \leftarrow \mathsf{TList} : (\mathbf{C}_0, \mathbf{C}_1)$
    $\mathsf{MList} \leftarrow \mathsf{MList} : (\mathbf{m}_0, \mathbf{m}_1)$
    $\mathbf{tk} \leftarrow_\$ \mathsf{FE.TGen}(\mathsf{msk}, \mathbf{C}_b)$
    $\mathbf{c} \leftarrow_\$ \mathsf{FE.Enc}(\mathsf{mpk}, \mathbf{m}_b)$
    return $(\mathbf{tk}, \mathbf{c}, z)$

$\underline{\mathrm{TGen}(\mathsf{C})}$:
    $\mathsf{TList} \leftarrow \mathsf{TList} : (\mathsf{C}, \mathsf{C})$
    $\mathsf{tk} \leftarrow_\$ \mathsf{FE.TGen}(\mathsf{msk}, \mathsf{C})$
    return $\mathsf{tk}$
```

Figure 5.1: Game defining full privacy of a functional encryption scheme FE.

shows that for (stateless) samplers the definition self-composes and we consider the simpler single-shot game in the remainder of the thesis.

RESTRICTED PRIV AND PRIV-TO. We call an adversary *token-only* if $\mathcal{S}$ does not output any messages, and call the resulting security notion PRIV-TO. Note that, for token-only adversaries, the additional legitimacy constraint above is redundant. We call an adversary *restricted* if for every second-phase TGEN query C_2 there is a first-phase TGEN query C_1 such that $\mathsf{C}_2(\mathbf{m}_b) = \mathsf{C}_1(\mathbf{m}_b)$ for $b \in \{0,1\}$. Intuitively, this amounts to imposing that images exposed via second-stage queries (i.e., those placed *after* receiving the challenge) can reveal no more than the images obtained in the first stage (i.e., from queries placed *before* receiving the challenge). We call the resulting security notion Res-PRIV. We emphasize that the Res-PRIV model inherits many of the strengths of the full PRIV model such as arbitrary correlations and a wide range of adaptive token queries.[2]

5.1 The case of keyword search

Two important aspects of our definition are that it considers (1) challenge keywords that do not match any of the encrypted messages and challenge messages that not match any of the keywords—we call these keywords and messages *unpaired*; and (2) *low-entropy* messages/keywords that are correlated with the high-entropy searches whose privacy must be protected. The former aspect entails that the full equality pattern of challenge messages and keywords may remain hidden from the adversary (and hence a wider class of non-

[2]When the restriction here is imposed on the IND-CPA model for point function, the resulting model remains as strong as the full IND-CPA model.

trivial attacks can be launched). Although the adversary always obtains the image matrix resulting from evaluating tokens on ciphertexts (and hence sees the equality pattern between *paired* challenge keywords and messages), the repetition patterns among unpaired messages or unpaired keywords is not necessarily leaked. In practice, this repetition pattern may reveal sensitive information as well. Low entropy messages and keywords model the presence of ciphertexts and tokens in the system, over which the uncertainty of the adversary may be small, but which are correlated with sensitive data that must still be protected. Indeed, our unpredictability notion allows the sampler to output such low-entropy keywords and messages as long as low-entropy keywords are equal on left and right. A real-or-random modeling of this setting cannot capture this scenario. When low-entropy messages differ on the left and right, the adversary cannot learn them via the TGen oracle due to the legitimacy condition: imposing that they are not leaked maps to IND-CPA security. When they are equal on the left and right, they can be learned by successive queries to the token extraction oracle, which permits capturing attack scenarios where *adaptive* searches over low entropy correlated messages may be carried out. In particular, this permits an adversary to recover a correlated repetition search pattern *after* the PRIV challenge has been revealed. As a result, low-entropy messages and keywords are tolerated, even when correlated with other messages or keywords. Furthermore, the values and equality patterns of high-entropy keywords are protected, as well as those of all encrypted messages for which a token was not explicitly extracted. Our main results in Sections 6.4 and 6.5 show the existence of keyword search schemes which are secure in the aforementioned scenarios.

5.2 On revealing images

The outputs of challenge circuits on challenge messages can be always computed by the adversary, and by imposing equality of images we ensure that they do not lead to trivial distinguishing attacks. (This is similar to the legitimacy condition in FE security models.) It is however less clear why these image values should be explicitly provide to the predictor in the unpredictability game, even when they are equal for left and right circuits-messages pairs. To see this, consider the sampler that for a random word w

outputs

$$\mathsf{w}_0 = \mathsf{w}, \quad \mathsf{w}_1 = \overline{\mathsf{w}}, \quad \mathsf{m}_{0,i} := \begin{cases} \mathsf{w} & \text{if } \mathsf{w}[i] = 1 \ ; \\ \overline{\mathsf{w}} & \text{otherwise,} \end{cases} \quad \text{and} \quad \mathsf{m}_{1,i} := \begin{cases} \overline{\mathsf{w}} & \text{if } \mathsf{w}[i] = 1 \ ; \\ \mathsf{w} & \text{otherwise.} \end{cases}$$

Note that $\mathsf{C}[\mathsf{w}_0](\mathsf{m}_{0,i}) = \mathsf{C}[\mathsf{w}_1](\mathsf{m}_{1,i}) = \mathsf{w}[i]$ and hence the images are equal on left and right. Word w_0 can be recovered bit by bit from the image values $\mathsf{C}[\mathsf{w}_b](\mathsf{m}_{0,i})$ and computing $1 - \mathsf{C}[\mathsf{w}_b](\mathsf{w}_0)$ would then reveal the challenge bit b. Finally, without access to the images $\mathsf{C}[\mathsf{w}_0](\mathsf{m}_{0,i})$ the sampler can be shown to be unpredictable as w is chosen randomly. On the other hand, in the presence of images, the sampler is trivially predicable. This counterexample is similar to that briefly discussed in [ATR14] and can be modified to show that the *enhanced* model of Boneh, Raghunathan and Segev [BRS13a] for the so-called $(k_1, \ldots, k_T)$-distributions is not achievable.

$$\begin{tikzcd}
 & & \text{ResPRIV} \arrow[dr] \arrow[dl, dashed] & \\
\text{PRIV} \arrow[r] & \text{ResPRIV} \wedge \text{CPA} \arrow[ur] \arrow[dr] & & \text{PRIV-TO} \\
 & & \text{PRIV-TO} \wedge \text{CPA} \arrow[ur] \arrow[d] & \\
 & & \text{CPA} &
\end{tikzcd}$$

Figure 5.2: Relations among security notions for private functional encryption. (The dotted implication only holds for keyword search schemes.)

5.3 Relations among notions

Clearly PRIV implies its weaker variant Res-PRIV, which in turn implies PRIV-TO. It is not too difficult to see that PRIV also implies IND-CPA.[3] A noteworthy consequence of this is that for *all-or-nothing* functionalities (such as PEKS, IBE or ABE) any PRIV-secure construction is also *index hiding* (aka. anonymous), whereby ciphertexts do not leak any information about their intended recipients (i.e., about tokens that may permit recovering the payload). Res-PRIV would imply a restricted analogue of IND-CPA

[3]Consider a sampler which does not output any circuits and simply returns (possibly low-entropy) messages included in the state *st* passed to it. This sampler is trivially unpredictable. Furthermore, the legitimacy conditions in the two games exactly match.

(where images in the second phase should match one in the first phase), which for point functions is equivalent to the standard IND-CPA model. IND-CPA security does not imply PRIV-TO: consider an IND-CPA-secure scheme that is modified to append circuits in the clear to their tokens. PRIV-TO does not imply IND-CPA either: consider a PRIV-TO-secure scheme that is modified to return messages in the clear with ciphertexts. (Note that these separations hold even for point functions.) Figure 5.2 summarizes relations among notions of security.

Chapter 6

Function Private Constructions

The results described in this chapter have been published in [ATR14, ABF16].

In this chapter, we present the second and main contribution of this thesis: five constructions of private functional encryption supporting different classes of functions and meeting varying degrees of security, namely (1) a white-box construction of an Anonymous IBE scheme based on composite-order groups, shown to be secure in the absence of correlated messages; (2) a simple and functionality-agnostic black-box construction from obfuscation, also shown to be secure in the absence of correlated messages; (3) a more evolved and still functionality-agnostic construction that achieves a form of function privacy that tolerates limited correlations between messages and functions; (4) a KS scheme achieving privacy in the presence of correlated messages beyond all previously proposed indistinguishability-based security definitions; (5) a KS construction that achieves our strongest notion of privacy (but relies on a more expressive form of obfuscation than the previous construction).

6.1 Two white-box constructions of AIBE schemes

We start by looking at Boyen and Waters [BW06] anonymous identity-based encryption scheme in the hope of showing that it already achieves some form of function privacy, as the decryption keys are randomized. Towards this end, we first present a simplified version of the original scheme and show that, in the random oracle model, not only IND-

CPA security is still guaranteed, we are also able to lift the selective-id constraint in the proof. Next, we show that the scheme is PRIV-TO secure up to two decryption keys. In fact, we also show that if the sampler outputs three keys, there is a trivial distinguishing attack. To improve security, we extend the scheme to groups of composite order and show that the extended version is PRIV-TO secure for an unbounded number of keys.

Remark. Using the anonymous identity-based encryption (AIBE) to public-key encryption with keyword search (KS) transform from [ABC+08], one can easily derive keyword search schemes where privacy guarantees of the keywords in KS are identical to those of identities in AIBE [ATR14].

6.1.1 A simplified Boyen-Waters AIBE scheme in the RO model

Here, we construct a new anonymous identity-based encryption scheme based on that of Boyen and Waters [BW06]. Our scheme relies on a bilinear groups of prime order. To eliminate the selective-id constraint of the original scheme, we replace identities with their hash values and model the hash function as a random oracle. Furthermore, we simplify the scheme by removing two group elements from the public parameters and from decryption keys, and obtain the final scheme in Figure 6.1. Compared to the original scheme, our scheme also saves two exponentiations in the key-extraction and encryption algorithms, and saves two pairing computations in the decryption algorithm. Our scheme preserves the original security properties, provided that the hash function h, sampled from the family $\mathcal{H}_\lambda : \mathsf{IdSp}_\lambda \to \mathbb{G}$, is modeled as a random oracle. Added to this, the scheme also has a weak form of token-only privacy where the sampler is (2,0)-bounded, i.e. the sampler outputs circuit-vectors with at most 2 identities and no correlated messages.

Theorem 2. *The anonymous identity-based encryption scheme* AIBE_1 *[Figure 6.1] is* IND-CPA *secure [Figure 2.5], in the random oracle model, assuming* DBDH *and* DLIN *are intractable [Definitions 1 and 2]. More precisely, for any adversary* $\mathcal{A}$ *in game* IND-CPA *against* AIBE_1*, there exists adversaries* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *such that*

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AIBE},\mathcal{A}}(\lambda) \leq q \cdot \mathbf{Adv}^{\mathsf{dbdh}}_{\mathcal{G}_\mathcal{P},\mathcal{B}_1}(\lambda) + q \cdot \mathbf{Adv}^{\mathsf{dlin}}_{\mathcal{G}_\mathcal{P},\mathcal{B}_2}(\lambda) \ ,$$

where q *is the number of queries* $\mathcal{A}$ *places to its random oracle.*

$\underline{\mathsf{AIBE.Setup}(1^\lambda)}$:	$\underline{\mathsf{AIBE.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})}$:	$\underline{\mathsf{AIBE.Enc}(\mathsf{mpk}, \mathsf{m}, \mathsf{id})}$:	$\underline{\mathsf{AIBE.Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{id}}, \mathsf{c})}$:
$\Gamma \leftarrow_\$ \mathcal{G}_\mathcal{P}(1^\lambda)$	$(\Gamma, \Omega, \mathsf{v}_1, \mathsf{v}_2, h) \leftarrow \mathsf{mpk}$	$(\Gamma, \Omega, \mathsf{v}_1, \mathsf{v}_2, h) \leftarrow \mathsf{mpk}$	$(\Gamma, \Omega, \mathsf{v}_1, \mathsf{v}_2, h) \leftarrow \mathsf{mpk}$
$(\mathsf{p}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}) \leftarrow \Gamma$	$(\mathsf{p}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}) \leftarrow \Gamma$	$(\mathsf{p}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}) \leftarrow \Gamma$	$(\mathsf{p}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}) \leftarrow \Gamma$
$\mathsf{w}, \mathsf{t}_1, \mathsf{t}_2 \leftarrow_\$ \mathbb{Z}_\mathsf{p}$	$(\mathsf{w}, \mathsf{t}_1, \mathsf{t}_2) \leftarrow \mathsf{msk}$	$\mathsf{s}, \mathsf{s}_1 \leftarrow_\$ \mathbb{Z}_\mathsf{p}$	$(\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2) \leftarrow \mathsf{sk}_{\mathsf{id}}$
$\Omega \leftarrow \mathbf{e}(\mathsf{g}, \mathsf{g})^{\mathsf{t}_1\mathsf{t}_2\mathsf{w}}$	$\mathsf{r} \leftarrow_\$ \mathbb{Z}_\mathsf{p}$	$\mathsf{h} \leftarrow h(\mathsf{id})$	$(\hat{\mathsf{c}}, \mathsf{c}_0, \mathsf{c}_1, \mathsf{c}_2) \leftarrow \mathsf{c}$
$\mathsf{v}_1 \leftarrow \mathsf{g}^{\mathsf{t}_1}$	$\mathsf{h} \leftarrow h(\mathsf{id})$	$\hat{\mathsf{c}} \leftarrow \Omega^{\mathsf{s}}\mathsf{m}$	$\mathsf{e}_0 \leftarrow \mathbf{e}(\mathsf{c}_0, \mathsf{d}_0)$
$\mathsf{v}_2 \leftarrow \mathsf{g}^{\mathsf{t}_2}$	$\mathsf{d}_0 \leftarrow \mathsf{g}^{\mathsf{r}\mathsf{t}_1\mathsf{t}_2}$	$\mathsf{c}_0 \leftarrow \mathsf{h}^{\mathsf{s}}$	$\mathsf{e}_1 \leftarrow \mathbf{e}(\mathsf{c}_1, \mathsf{d}_1)$
$h \leftarrow_\$ \mathcal{H}_\lambda$	$\mathsf{d}_1 \leftarrow \mathsf{g}^{-\mathsf{w}\mathsf{t}_2} \cdot \mathsf{h}^{-\mathsf{r}\mathsf{t}_2}$	$\mathsf{c}_1 \leftarrow \mathsf{v}_1^{\mathsf{s}-\mathsf{s}_1}$	$\mathsf{e}_2 \leftarrow \mathbf{e}(\mathsf{c}_2, \mathsf{d}_2)$
$\mathsf{mpk} \leftarrow (\Gamma, \Omega, \mathsf{v}_1, \mathsf{v}_2, h)$	$\mathsf{d}_2 \leftarrow \mathsf{g}^{-\mathsf{w}\mathsf{t}_1} \cdot \mathsf{h}^{-\mathsf{r}\mathsf{t}_1}$	$\mathsf{c}_2 \leftarrow \mathsf{v}_2^{\mathsf{s}_1}$	$\mathsf{m} \leftarrow \hat{\mathsf{c}} \cdot \mathsf{e}_0 \cdot \mathsf{e}_1 \cdot \mathsf{e}_2$
$\mathsf{msk} \leftarrow (\mathsf{w}, \mathsf{t}_1, \mathsf{t}_2)$	$\mathsf{sk}_{\mathsf{id}} \leftarrow (\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2)$	$\mathsf{c} \leftarrow (\hat{\mathsf{c}}, \mathsf{c}_0, \mathsf{c}_1, \mathsf{c}_2)$	return m
return $(\mathsf{mpk}, \mathsf{msk})$	return $\mathsf{sk}_{\mathsf{id}}$	return c	

Figure 6.1: AIBE_1, a simplified version of Boyen-Waters AIBE scheme.

Proof. The proof proceeds via a sequence of two game hops as follows.

Game$_0$: This game is identical to IND-CPA.

Game$_1$: In this game, we set $\hat{c} \leftarrow_\$ \mathbb{G}_\mathsf{T}$ instead of computing its value.

Game$_2$: We set $\mathsf{c}_0 \leftarrow_\$ \mathbb{G}$. Here, the challenge ciphertext does not depend on the challenge bit, therefore the advantage is 0.

We now reduce the distance between the games to the intractability of DBDH and DLIN assumptions.

Game$_0$ TO Game$_1$. Any adversary $\mathcal{A}$ with visible advantage difference in these two games can be converted to an adversary $\mathcal{B}_1$ that breaks the DBDH assumption. More precisely, we construct an adversary $\mathcal{B}_1$ [Figure 6.2] that interpolates between Game_0 and Game_1 by playing game $\mathsf{DBDH}^{\mathcal{B}_1}_{\mathcal{G}_\mathcal{P}}(1^\lambda)$ [Figure 2.1]. The hash function h is modeled as a random oracle and we assume, without loss of generality, that $\mathcal{A}$ always asks for the hash value of id before querying id to oracles TGen or LR. Furthermore, for simplicity of exposition, we assume that $\mathcal{A}$ places at most q queries to the random oracle, where q is bounded by some polynomial in the security parameter λ, since $\mathcal{A}$ is required to run in polynomial-time. $\mathcal{B}_1$ randomly tries to guess which query $\mathsf{i} \in \{0, q\text{-}1\}$ contains id_b on which adversary $\mathcal{A}$ asks to be challenged. When i is *not* successfully guessed, $\mathcal{B}_1$ simply aborts. But when it is, which happens with probability $\frac{1}{q}$, $\mathcal{B}_1$ perfectly simulates Game_0 if Z is of the form

$\mathbf{e}(\mathsf{g},\mathsf{g})^{\mathsf{z}_1\mathsf{z}_2\mathsf{z}_3}$, and perfectly simulates Game_1 if Z is just a random element in $\mathbb{G}_\mathsf{T}$. Notice that this implies $\mathsf{w} = \mathsf{z}_1\mathsf{z}_2$ and $\mathsf{s} = \mathsf{z}_3$.

Random function h is consistently computed: if queried twice on the same id, the same result is returned. When $\mathcal{B}_1$ successfully completes its execution, the function is set to $(\mathsf{g}^{\mathsf{z}_1})^{\mathsf{x}}$ for every id but id_b, and to g^{x} in this particular case, where x is a random value sampled from $\mathbb{Z}_\mathsf{p}$.

Challenge is well formed, as well as secret keys for random exponents $\mathsf{r}' = \frac{\mathsf{r}-\mathsf{z}_2}{\mathsf{x}}$, where x here is the value used to compute the hash of the corresponding id and r is sampled from $\mathbb{Z}_\mathsf{p}$. For completeness, we present the equalities between the original expressions and those computed by $\mathcal{B}_1$. For $\hat{c}$ we used the case where $\mathsf{Z} = \mathbf{e}(\mathsf{g},\mathsf{g})^{\mathsf{z}_1\mathsf{z}_2\mathsf{z}_3}$, which corresponds to the simulation of Game_0.

$$\mathsf{d}_0 = \mathsf{g}^{\mathsf{r}'\mathsf{t}_1\mathsf{t}_2} = \mathsf{g}^{\frac{\mathsf{r}-\mathsf{z}_2}{\mathsf{x}}\mathsf{t}_1\mathsf{t}_2} = \mathsf{g}^{\frac{\mathsf{r}\mathsf{t}_1\mathsf{t}_2}{\mathsf{x}}} \cdot \mathsf{g}^{\frac{-\mathsf{z}_2\mathsf{t}_1\mathsf{t}_2}{\mathsf{x}}} = \mathsf{g}^{\frac{\mathsf{r}\mathsf{t}_1\mathsf{t}_2}{\mathsf{x}}} \cdot \mathsf{Z}_2^{\frac{-\mathsf{t}_1\mathsf{t}_2}{\mathsf{x}}}$$
$$\mathsf{d}_1 = \mathsf{g}^{-\mathsf{w}\mathsf{t}_2} \cdot \mathsf{h}^{-\mathsf{r}'\mathsf{t}_2} = \mathsf{g}^{-\mathsf{z}_1\mathsf{z}_2\mathsf{t}_2} \cdot [(\mathsf{g}^{\mathsf{z}_1})^{\mathsf{x}}]^{-\frac{\mathsf{r}-\mathsf{z}_2}{\mathsf{x}}\mathsf{t}_2} = \mathsf{Z}_1^{-\mathsf{r}\mathsf{t}_2}$$
$$\mathsf{d}_2 = \mathsf{g}^{-\mathsf{w}\mathsf{t}_1} \cdot \mathsf{h}^{-\mathsf{r}'\mathsf{t}_1} = \mathsf{g}^{-\mathsf{z}_1\mathsf{z}_2\mathsf{t}_1} \cdot [(\mathsf{g}^{\mathsf{z}_1})^{\mathsf{x}}]^{-\frac{\mathsf{r}-\mathsf{z}_2}{\mathsf{x}}\mathsf{t}_1} = \mathsf{Z}_1^{-\mathsf{r}\mathsf{t}_1}$$

$$\hat{\mathsf{c}} = \Omega^{\mathsf{s}} \cdot \mathsf{m} = [\mathbf{e}(\mathsf{g},\mathsf{g})^{\mathsf{t}_1\mathsf{t}_2\mathsf{w}}]^{\mathsf{s}} \cdot \mathsf{m} = [\mathbf{e}(\mathsf{g},\mathsf{g})^{\mathsf{t}_1\mathsf{t}_2\mathsf{z}_1\mathsf{z}_2}]^{\mathsf{z}_3} \cdot \mathsf{m} = \mathsf{Z}^{\mathsf{t}_1\mathsf{t}_2} \cdot \mathsf{m}$$
$$\mathsf{c}_0 = \mathsf{h}^{\mathsf{s}} = (\mathsf{g}^{\mathsf{x}})^{\mathsf{z}_3} = \mathsf{Z}_3^{\mathsf{x}}$$
$$\mathsf{c}_1 = \mathsf{v}_1^{\mathsf{s}-\mathsf{s}_1} = (\mathsf{g}_1^{\mathsf{t}})^{\mathsf{z}_3-\mathsf{s}_1} = (\mathsf{Z}_3 \cdot g^{-\mathsf{s}_1})^{\mathsf{t}_1}$$
$$\mathsf{c}_2 = \mathsf{v}_2^{\mathsf{s}_1}$$

Therefore, we have that $\Pr[\mathrm{Game}_0(1^\lambda)] - \Pr[\mathrm{Game}_1(1^\lambda)] \leq q \cdot \mathbf{Adv}^{\mathsf{dbdh}}_{\mathcal{G}_\mathcal{P},\mathcal{B}_1}(\lambda)$.

Game$_1$ TO Game$_2$. For $\mathcal{A}$ to successfully distinguish between Game_1 and Game_2, the DLIN assumption would have to be tractable. Formally, we show this by constructing an algorithm $\mathcal{B}_2$ [Figure 6.3] that interpolates between Game_1 and Game_2 by playing game $\mathsf{DLIN}^{\mathcal{B}_2}_{\mathcal{G}_\mathcal{P}}(1^\lambda)$ [Figure 2.1]. As before, the hash function h is modeled as a random oracle to which $\mathcal{A}$ places at most q queries and we assume that $\mathcal{A}$ always asks for the hash value of id before querying it to TGEN or LR oracles. Employing the same strategy as before, algorithm $\mathcal{B}_2$ randomly tries to guess which query $\mathsf{i} \in \{0, q\text{-}1\}$ contains id_b. When i is successfully guessed, which happens with probability at least $\frac{1}{q}$, $\mathcal{B}_2$ perfectly simulates Game_1 if Z is of the form $\mathsf{g}^{\mathsf{z}_3+\mathsf{z}_4}$, and perfectly simulates Game_2 otherwise. This implies

```
$\underline{\mathcal{B}_1(\Gamma, \mathsf{Z}_1, \mathsf{Z}_2, \mathsf{Z}_3, \mathsf{Z})}$:
$\mathsf{t}_1, \mathsf{t}_2 \leftarrow_\$ \mathbb{Z}_\mathsf{p}$
$\Omega \leftarrow \mathbf{e}(\mathsf{Z}_1, \mathsf{Z}_2)^{\mathsf{t}_1\mathsf{t}_2}$
$\mathsf{v}_1 \leftarrow \mathsf{g}^{\mathsf{t}_1}$
$\mathsf{v}_2 \leftarrow \mathsf{g}^{\mathsf{t}_2}$
$\mathsf{mpk} \leftarrow (\Omega, \mathsf{v}_1, \mathsf{v}_2)$
$counter \leftarrow 0$
$i^\star \leftarrow_\$ \{0, q\text{-}1\}$
$b \leftarrow_\$ \{0, 1\}$
$b' \leftarrow_\$ \mathcal{A}^{\mathrm{H},\mathrm{TGen},\mathrm{LR}}(\mathsf{mpk})$
return $(b = b')$

$\underline{\mathrm{H}(\mathsf{id})}$:
$(\mathsf{x}, i) \leftarrow \mathsf{list}[\mathsf{id}]$
if $(\mathsf{x} = \perp)$
    $\mathsf{x} \leftarrow_\$ \mathbb{Z}_\mathsf{p}$
    $i \leftarrow counter$
    $counter \leftarrow counter + 1$
    $\mathsf{list}[\mathsf{id}] \leftarrow (\mathsf{x}, i)$
if $(i = i^\star)$ $\mathsf{h} \leftarrow \mathsf{g}^\mathsf{x}$
else $\mathsf{h} \leftarrow \mathsf{Z}_1^\mathsf{x}$
return $\mathsf{h}$

$\underline{\mathrm{TGen}(\mathsf{id})}$:
$(\mathsf{x}, i) \leftarrow \mathsf{list}[\mathsf{id}]$
if $(i = i^\star)$ abort
$\mathsf{r} \leftarrow_\$ \mathbb{Z}_\mathsf{p}$
$\mathsf{d}_0 \leftarrow \mathsf{g}^{\frac{\mathsf{r}\mathsf{t}_1\mathsf{t}_2}{\mathsf{x}}} \cdot \mathsf{Z}_2^{\frac{-\mathsf{t}_1\mathsf{t}_2}{\mathsf{x}}}$
$\mathsf{d}_1 \leftarrow \mathsf{Z}_1^{-\mathsf{r}\mathsf{t}_2}$
$\mathsf{d}_2 \leftarrow \mathsf{Z}_1^{-\mathsf{r}\mathsf{t}_1}$
$\mathsf{sk}_\mathsf{id} \leftarrow (\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2)$
return $\mathsf{sk}_\mathsf{id}$

$\underline{\mathrm{LR}((\mathsf{id}_0, \mathsf{m}_0), (\mathsf{id}_1, \mathsf{m}_1))}$:
$(\mathsf{x}, i) \leftarrow \mathsf{list}[\mathsf{id}_b]$
if $(i \neq i^\star)$ abort
$\mathsf{s}_1 \leftarrow_\$ \mathbb{Z}_\mathsf{p}$
$\hat{\mathsf{c}} \leftarrow \mathsf{Z}^{\mathsf{t}_1\mathsf{t}_2} \cdot \mathsf{m}_b$
$\mathsf{c}_0 \leftarrow \mathsf{Z}_3^\mathsf{x}$
$\mathsf{c}_1 \leftarrow \mathsf{Z}_3^{\mathsf{t}_1} \cdot \mathsf{g}^{-\mathsf{t}_1\mathsf{s}_1}$
$\mathsf{c}_2 \leftarrow \mathsf{v}_2^{\mathsf{s}_1}$
$\mathsf{c} \leftarrow (\hat{\mathsf{c}}, \mathsf{c}_0, \mathsf{c}_1, \mathsf{c}_2)$
return $\mathsf{c}$
```

Figure 6.2: DBDH adversary $\mathcal{B}_1$, as part of proof of Theorem 2.

that $\mathsf{t}_1 = \mathsf{z}_1$ and $\mathsf{t}_2 = \mathsf{z}_2$. Random function h sets $(\mathsf{g}^{\mathsf{z}_2})^\mathsf{x}$ for every id but the id_b, which is set to g^x, where x is a random value sampled from $\mathbb{Z}_\mathsf{p}$. Challenge is well formed, as well as secret keys for random exponents $\mathsf{r}' = \frac{\mathsf{r}}{\mathsf{z}_2}$, where r is sampled from $\mathbb{Z}_\mathsf{p}$. Finally, notice that $\mathsf{s} = \mathsf{z}_3 + \mathsf{z}_4$ and $\mathsf{s}_1 = \mathsf{z}_4$, and that t_1, t_2, r', s and s_1 are uniformly distributed over $\mathbb{Z}_\mathsf{p}$, as they should be. For completeness, we present the equalities between the original expressions and those computed by $\mathcal{B}_2$. For c_0 we used the case where $\mathsf{Z} = \mathsf{g}^{\mathsf{z}_3+\mathsf{z}_4}$, which corresponds to the simulation of Game_1.

$$\mathsf{d}_0 = \mathsf{g}^{\mathsf{r}'\mathsf{t}_1\mathsf{t}_2} = \mathsf{g}^{\frac{\mathsf{r}}{\mathsf{z}_2}\mathsf{z}_1\mathsf{z}_2} = (\mathsf{g}^{\mathsf{z}_1})^\mathsf{r} = \mathsf{Z}_1^\mathsf{r}$$
$$\mathsf{d}_1 = \mathsf{g}^{-\mathsf{w}\mathsf{t}_2} \cdot \mathsf{h}^{-\mathsf{r}'\mathsf{t}_2} = \mathsf{g}^{-\mathsf{w}\mathsf{z}_2} \cdot [(\mathsf{g}^{\mathsf{z}_2})^\mathsf{x}]^{-\frac{\mathsf{r}}{\mathsf{z}_2}\mathsf{z}_2} = \mathsf{Z}_2^{-\mathsf{w}} \cdot \mathsf{Z}_2^{-\mathsf{x}\mathsf{r}} = \mathsf{Z}_2^{-\mathsf{w}-\mathsf{x}\mathsf{r}}$$
$$\mathsf{d}_2 = \mathsf{g}^{-\mathsf{w}\mathsf{t}_1} \cdot \mathsf{h}^{-\mathsf{r}'\mathsf{t}_1} = \mathsf{g}^{-\mathsf{w}\mathsf{z}_1} \cdot [(\mathsf{g}^{\mathsf{z}_2})^\mathsf{x}]^{-\frac{\mathsf{r}}{\mathsf{z}_2}\mathsf{z}_1} = \mathsf{Z}_1^{-\mathsf{w}} \cdot \mathsf{Z}_1^{-\mathsf{x}\mathsf{r}} = \mathsf{Z}_1^{-\mathsf{w}-\mathsf{x}\mathsf{r}}$$

$$\mathsf{c}_0 = \mathsf{h}^\mathsf{s} = (\mathsf{g}^\mathsf{x})^{\mathsf{z}_3+\mathsf{z}_4} = \mathsf{Z}^\mathsf{x}$$
$$\mathsf{c}_1 = \mathsf{v}_1^{\mathsf{s}-\mathsf{s}_1} = (\mathsf{g}^{\mathsf{z}_1})^{(\mathsf{z}_3+\mathsf{z}_4)-\mathsf{z}_4} = \mathsf{Z}_{13}$$
$$\mathsf{c}_2 = \mathsf{v}_2^{\mathsf{s}_1} = (\mathsf{g}^{\mathsf{z}_2})^{\mathsf{z}_4} = \mathsf{Z}_{24}$$

Therefore, we have that $\Pr[\mathrm{Game}_1(1^\lambda)] - \Pr[\mathrm{Game}_2(1^\lambda)] \leq \mathsf{q} \cdot \mathbf{Adv}^{\mathsf{dlin}}_{\mathcal{G}_\mathcal{P},\mathcal{B}_2}(\lambda)$. In Game_2, the challenge ciphertext is independent of the challenge bit b, which concludes our proof.

□

```
$\mathcal{B}_2(\Gamma, Z_1, Z_2, Z_{13}, Z_{24}, Z)$:
    $w \leftarrow_\$ \mathbb{Z}_p$
    $\Omega \leftarrow \mathbf{e}(Z_1, Z_2)^w$
    $v_1 \leftarrow Z_1;\ v_2 \leftarrow Z_2$
    $mpk \leftarrow (\Omega, v_1, v_2)$
    $i \leftarrow_\$ \{0, q\text{-}1\}$
    $b \leftarrow_\$ \{0,1\}$
    $b' \leftarrow_\$ \mathcal{A}^{H,\text{TGen},\text{LR}}(mpk)$
    return $(b = b')$

$H(id)$:
    $(x, i) \leftarrow list[id]$
    if $(x = \perp)$
        $x \leftarrow_\$ \mathbb{Z}_p$
        $i \leftarrow counter$
        $counter \leftarrow counter + 1$
        $list[id] \leftarrow (x, i)$
    if $(i = i^\star)$ $h \leftarrow g^x$
    else $h \leftarrow Z_2^x$
    return $h$

$\text{TGen}(id)$:
    $(x, i) \leftarrow list[id]$
    if $(i = i^\star)$ abort
    $r \leftarrow_\$ \mathbb{Z}_p$
    $d_0 \leftarrow Z_1^r$
    $d_1 \leftarrow Z_2^{-w-xr}$
    $d_2 \leftarrow Z_1^{-w-xr}$
    $sk_{id} \leftarrow (d_0, d_1, d_2)$
    return $sk_{id}$

$\text{LR}((id_0, m_0), (id_1, m_1))$:
    $(x, i) \leftarrow list[id_b]$
    if $(i \neq i^\star)$ abort
    $\hat{c} \leftarrow_\$ \mathbb{G}_T$
    $c_0 \leftarrow Z^x$
    $c_1 \leftarrow Z_{13}$
    $c_2 \leftarrow Z_{24}$
    $c \leftarrow (\hat{c}, c_0, c_1, c_2)$
    return $c$
```

Figure 6.3: DLIN adversary $\mathcal{B}_2$, as part of proof of Theorem 2.

Besides being IND-CPA secure, we show that scheme AIBE_1 also has a weak form of token-only privacy against unpredictable samplers outputting at most 2 identities and no correlated messages.

Theorem 3. *The anonymous identity-based encryption scheme* AIBE_1 *[Figure 6.1] is* PRIV-TO *secure against unpredictable samplers that are* $(2,0)$*-bounded and functionally-differing, in the random oracle model, assuming DLIN is intractable [Definition 2].*

Proof. Let $(\mathcal{S}, \mathcal{A})$ be a legitimate adversary in game $\text{PRIV-TO}^{\mathcal{S},\mathcal{A}}_{\mathsf{AIBE}_1}$ [Figure 5.1]. $\mathcal{S}$ is $(2,0)$-bounded and functionally-differing. The proof proceeds as a sequence of two games hops.

Game_0 : This game is identical to PRIV-TO game.

Game_1 : In this game, we set a bad event in case the adversary $\mathcal{A}$ queries to the random oracle any of the identities output by $\mathcal{S}$.

Game_2 : Independently of the bit b, we always extract the challenge decryption keys as if $b = 1$. Since the challenge keys do not depend on the challenge bit, the advantage here is 0.

We now analyse the distance between the games.

Game_0 TO Game_1. An adversary $\mathcal{A}$ that triggers the bad even can easily be used by a predictor $\mathcal{P}$ to break the unpredictability of $\mathcal{S}$, as asking for any of the identities output by $\mathcal{S}$, which is functionally-differing, amounts to break its unpredictability. Therefore, $\Pr[\text{Game}_0(1^\lambda)] - \Pr[\text{Game}_1(1^\lambda)] \leq \mathbf{Adv}^{\text{pred}}_{\mathcal{S},\mathcal{P}}(\lambda)$.

Game$_1$ TO Game$_2$. Let $\mathbf{id}_0$ and $\mathbf{id}_1$ be the two challenge vectors output by $\mathcal{S}$, each containing at most two identities–recall that $\mathcal{S}$ is $(2,0)$-bounded. It can either be that:

1. $\mathbf{id}_0[0] \neq \mathbf{id}_0[1] \wedge \mathbf{id}_1[0] \neq \mathbf{id}_1[1]$
2. $\mathbf{id}_0[0] = \mathbf{id}_0[1] \wedge \mathbf{id}_1[0] = \mathbf{id}_1[1]$
3. $\mathbf{id}_0[0] = \mathbf{id}_0[1] \wedge \mathbf{id}_1[0] \neq \mathbf{id}_1[1]$
4. $\mathbf{id}_0[0] \neq \mathbf{id}_0[1] \wedge \mathbf{id}_1[0] = \mathbf{id}_1[1]$

From the adversary's point of view, Case 1 and Case 2 are identical in Game$_1$ and Game$_2$, as the adversary has no access to the hash values of any of the identities. Case 3 and Case 4 are symmetric.

By building an adversary $\mathcal{B}_1$ [Figure 6.4] that plays game $\mathsf{DLIN}^{\mathcal{B}_1}_{\mathcal{G}_\mathcal{P}}(1^\lambda)$ [Figure 2.1] and simulates game Game$_1$ in such a way that $\mathcal{A}$'s guess can be forward to game $\mathsf{DLIN}^{\mathcal{B}_1}_{\mathcal{G}_\mathcal{P}}(1^\lambda)$, we upper-bound the distance between Game$_1$ and Game$_2$ to the hardness of deciding on an instance of this problem.

The master secret key is set as following: $\mathsf{t}_1 = \mathsf{z}_1$, $\mathsf{t}_2 = \mathsf{z}_1 \cdot \mathsf{a}$ for random $\mathsf{a} \in \mathbb{Z}_\mathsf{p}$, and $\mathsf{w} = \frac{\mathsf{z}_3 \cdot \mathsf{b}}{\mathsf{z}_1}$ for random $\mathsf{b} \in \mathbb{Z}_\mathsf{p}$. Although the values of t_1, t_2 and w are unknown to $\mathcal{B}_1$, the corresponding public parameters can still be consistently computed:

$$\Omega = \mathbf{e}(\mathsf{g},\mathsf{g})^{\mathsf{t}_1\mathsf{t}_2\mathsf{w}} = \mathbf{e}(\mathsf{g},\mathsf{g})^{\mathsf{z}_1\mathsf{z}_1\mathsf{a}\frac{\mathsf{z}_3\cdot\mathsf{b}}{\mathsf{z}_1}} = \mathbf{e}(\mathsf{Z}_{13},\mathsf{g})^{\mathsf{ab}}$$
$$\mathsf{v}_1 = \mathsf{g}^{\mathsf{t}_1} = \mathsf{Z}_1$$
$$\mathsf{v}_2 = \mathsf{g}^{\mathsf{t}_2} = (\mathsf{Z}_1)^{\mathsf{a}}$$

The hash function H is modeled as a random oracle and set to $(\mathsf{g}^{\mathsf{z}_1})^{\mathsf{x}} \cdot \mathsf{g}^{-\frac{1}{\mathsf{y}}}$, for random $(\mathsf{x},\mathsf{y}) \in \mathbb{Z}_\mathsf{p}^2$. We assume, without loss of generality, that $\mathcal{A}$ always asks for the hash value of id before querying id to oracle TGen. Whenever asked to extract a private key on some id, we set $\mathsf{r} = \mathsf{w} \cdot \mathsf{y}$, where y is the value used to compute the hash of that particular id. Note that this still makes r uniformly distributed over $\mathbb{Z}_\mathsf{p}$ and independent of h and w. Given this, private keys can be extracted as follows:

$$d_0 = g^{rt_1t_2} = g^{wyt_1t_2} = g^{\frac{z_3\cdot b}{z_1}yz_1z_1a} = (Z_{13})^{aby}$$
$$d_1 = g^{-wt_2}\cdot h^{-rt_2} = g^{-wt_2}\cdot[(g^{z_1})^x\cdot g^{-\frac{1}{y}}]^{-wyt_2} = g^{-z_1xwyt_2} = g^{-z_1x\frac{z_3\cdot b}{z_1}yz_1a} = (Z_{13})^{-abxy}$$
$$d_2 = g^{-wt_1}\cdot h^{-rt_1} = g^{-wt_1}\cdot[(g^{z_1})^x\cdot g^{-\frac{1}{y}}]^{-wyt_1} = g^{-z_1xwyt_1} = g^{-z_1x\frac{z_3\cdot b}{z_1}yz_1} = (Z_{13})^{-bxy}$$

To complete the simulation, we extract two private keys to challenge $\mathcal{A}$, such that these private keys are for the same id if $\mathcal{B}_1$ received a valid DLIN tuple, and for different ids otherwise. Let $sk^\star = (d_0^\star, d_1^\star, d_2^\star)$ and $sk^\circ = (d_0^\circ, d_1^\circ, d_2^\circ)$ be the challenge keys. We set $h = g^{z_1z_4}$, $r^\star = \frac{b}{(z_1)^2}$ and $r^\circ = \frac{z_2+b}{(z_1)^2}$. Note that h is uniformly distributed over $\mathbb{G}$, and $r^\star$ and r° are uniformly distributed over $\mathbb{Z}_p$, independent of each other and of w. For completeness, we present the equalities between the original expressions and those computed by $\mathcal{B}_1$:

$$d_0^\star = g^{r^\star t_1t_2} = g^{\frac{b}{(z_1)^2}z_1z_1a} = g^{ab}$$
$$d_1^\star = g^{-wt_2}\cdot h^{-r^\star t_2} = g^{-\frac{z_3b}{z_1}z_1a}\cdot(g^{z_1z_4})^{-\frac{b}{(z_1)^2}z_1a} = (g^{-ab})^{z_3}\cdot(g^{-ab})^{z_4} = Z^{-ab}$$
$$d_1^\star = g^{-wt_1}\cdot h^{-r^\star t_1} = g^{-\frac{z_3b}{z_1}z_1}\cdot(g^{z_1z_4})^{-\frac{b}{(z_1)^2}z_1} = (g^{-b})^{z_3}\cdot(g^{-b})^{z_4} = Z^{-b}$$

$$d_0^\circ = g^{r^\circ t_1t_2} = g^{\frac{z_2+b}{(z_1)^2}z_1z_1\cdot a} = g^{z_2\cdot a+ab} = (Z_2)^a\cdot g^{ab}$$
$$d_1^\circ = g^{-wt_2}\cdot h^{-r^\circ t_2} = g^{-\frac{z_3b}{z_1}z_1a}\cdot(g^{z_1z_4})^{-\frac{z_2+b}{(z_1)^2}z_1a} = (g^{-ab})^{(z_3+z_4)}\cdot(g^{z_2z_4})^{-a} = Z^{-ab}\cdot(Z_{24})^{-a}$$
$$d_2^\circ = g^{-wt_1}\cdot h^{-r^\circ t_1} = g^{-\frac{z_3b}{z_1}z_1}\cdot(g^{z_1z_4})^{-\frac{z_2+b}{(z_1)^2}z_1} = (g^{-b})^{(z_3+z_4)}\cdot(g^{z_2z_4})^{-1} = Z^{-b}\cdot(Z_{24})^{-1}$$

Therefore, we have that

$$\Pr[\mathrm{Game}_1(1^\lambda)] - \Pr[\mathrm{Game}_2(1^\lambda)] \leq \Pr[\mathrm{Game}_1(1^\lambda)] - \frac{1}{2} = \frac{1}{2}\cdot\mathbf{Adv}^{\mathsf{dlin}}_{\mathcal{G}_\mathcal{P},\mathcal{B}_1}(\lambda).$$

To conclude our proof:

$$\mathbf{Adv}^{\text{priv-to}}_{\mathsf{AIBE}_1,(\mathcal{S},\mathcal{A})}(\lambda) \leq 2\cdot\mathbf{Adv}^{\text{pred}}_{\mathcal{S},\mathcal{P}}(\lambda) + \mathbf{Adv}^{\mathsf{dlin}}_{\mathcal{G}_\mathcal{P},\mathcal{B}_1}(\lambda).$$

$\square$

One might wonder why AIBE_1 is function private only up to two keys, and why we cannot extend the result by applying a standard hybrid argument [BBM00]. We answer this questions by means of a concrete attack against scheme.

```
$\underline{\mathcal{B}_1(\Gamma, \mathsf{Z}_1, \mathsf{Z}_2, \mathsf{Z}_{13}, \mathsf{Z}_{24}, \mathsf{Z})}$:
$\mathsf{a} \leftarrow_{\$} \mathbb{Z}_p, \mathsf{b} \leftarrow_{\$} \mathbb{Z}_p$
$\Omega \leftarrow \mathbf{e}(\mathsf{Z}_{13}, \mathsf{g})^{\mathsf{ab}}$
$\mathsf{v}_1 \leftarrow \mathsf{Z}_1$
$\mathsf{v}_2 \leftarrow (\mathsf{Z}_1)^{\mathsf{a}}$
$\mathsf{mpk} \leftarrow (\Omega, \mathsf{v}_1, \mathsf{v}_2)$
$b' \leftarrow_{\$} \mathcal{A}^{\mathrm{H},\mathrm{TGen},\mathrm{LR}}(\mathsf{mpk})$
if $(\mathbf{id}_0[0] = \mathbf{id}_0[1])$ return $b'$
else return $\neg b'$
```

```
$\underline{\mathrm{H}(\mathsf{id})}$:
$(\mathsf{x}, \mathsf{y}) \leftarrow \mathsf{list}[\mathsf{id}]$
if $((\mathsf{x}, \mathsf{y}) = \perp)$
    $\mathsf{x} \leftarrow_{\$} \mathbb{Z}_p$
    $\mathsf{y} \leftarrow_{\$} \mathbb{Z}_p$
    $\mathsf{list}[\mathsf{id}] \leftarrow (\mathsf{x}, \mathsf{y})$
$\mathsf{h} \leftarrow (\mathsf{g}^{\mathsf{z}_1})^{\mathsf{x}} \cdot \mathsf{g}^{-\frac{1}{\mathsf{y}}}$
return $\mathsf{h}$
```

```
$\underline{\mathrm{TGen}(\mathsf{id})}$:
$(\mathsf{x}, \mathsf{y}) \leftarrow \mathsf{list}[\mathsf{id}]$
$\mathsf{d}_0 \leftarrow (\mathsf{Z}_{13})^{\mathsf{aby}}$
$\mathsf{d}_1 \leftarrow (\mathsf{Z}_{13})^{-\mathsf{abxy}}$
$\mathsf{d}_2 \leftarrow (\mathsf{Z}_{13})^{-\mathsf{bxy}}$
$\mathsf{sk}_{\mathsf{id}} \leftarrow (\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2)$
return $\mathsf{sk}_{\mathsf{id}}$
```

```
$\underline{\mathrm{LR}(st)}$:
$(\mathbf{id}_0, \mathbf{id}_1) \leftarrow_{\$} \mathcal{S}(st)$
$\mathsf{d}_0^\star \leftarrow \mathsf{g}^{\mathsf{ab}}$
$\mathsf{d}_1^\star \leftarrow \mathsf{Z}^{-\mathsf{ab}}$
$\mathsf{d}_2^\star \leftarrow \mathsf{Z}^{-\mathsf{b}}$
$\mathsf{sk}_0 \leftarrow (\mathsf{d}_0^\star, \mathsf{d}_1^\star, \mathsf{d}_2^\star)$
$\mathsf{d}_0^\circ \leftarrow (\mathsf{Z}_2)^{\mathsf{a}} \cdot \mathsf{g}^{\mathsf{ab}}$
$\mathsf{d}_1^\circ \leftarrow \mathsf{Z}^{-\mathsf{ab}} \cdot (\mathsf{Z}_{24})^{-\mathsf{a}}$
$\mathsf{d}_2^\circ \leftarrow \mathsf{Z}^{-\mathsf{b}} \cdot (\mathsf{Z}_{24})^{-1}$
$\mathsf{sk}_1 \leftarrow (\mathsf{d}_0^\circ, \mathsf{d}_1^\circ, \mathsf{d}_2^\circ)$
return $(\mathsf{sk}_0, \mathsf{sk}_1)$
```

Figure 6.4: DLIN adversary $\mathcal{B}_1$, as part of proof of Theorem 3.

Suppose we have a $(3, 0)$-bounded sampler $\mathcal{S}$ that samples id_0 and id_1 uniformly at random from IdSp_λ and outputs $((\mathsf{id}_0, \mathsf{id}_0, \mathsf{id}_0), (\mathsf{id}_0, \mathsf{id}_0, \mathsf{id}_1))$. $\mathcal{S}$ is clearly unpredictable. Let $(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{sk}_2)$ be the challenge decryption keys that adversary $\mathcal{A}$ receives. The goal of $\mathcal{A}$ is to decide whether all these keys are associated with the same identity or not. We further expand sk_i to $(\mathsf{d}_{i0}, \mathsf{d}_{i1}, \mathsf{d}_{i2})$ according to our scheme. If the keys were generated honestly, i.e. by following the algorithm $\mathsf{AIBE}_1.\mathsf{KeyGen}()$ as described in Figure 6.1, the adversary simply has to check if

$$\mathbf{e}\left(\frac{\mathsf{d}_{10}}{\mathsf{d}_{00}}, \frac{\mathsf{d}_{21}}{\mathsf{d}_{01}}\right) \overset{?}{=} \mathbf{e}\left(\frac{\mathsf{d}_{00}}{\mathsf{d}_{20}}, \frac{\mathsf{d}_{01}}{\mathsf{d}_{11}}\right)$$

to determine the form of the tuple. If the equality is true, then the three secret keys have been extracted from the same id[1]. If the result is false, then keys have been extracted from $(\mathsf{id}_0, \mathsf{id}_0, \mathsf{id}_1)$. For completeness, we show this by expanding and simplifying the

[1]Collisions in the hash function h may lead to false positive results but only occur with negligible probability.

above expression.

$$\begin{aligned}
\mathbf{e}(\frac{\mathsf{d}_{10}}{\mathsf{d}_{00}}, \frac{\mathsf{d}_{21}}{\mathsf{d}_{01}}) &= \mathbf{e}(\frac{\mathsf{d}_{00}}{\mathsf{d}_{20}}, \frac{\mathsf{d}_{01}}{\mathsf{d}_{11}}) \Leftrightarrow \\
\mathbf{e}(\frac{g^{r_1 t_1 t_2}}{g^{r_0 t_1 t_2}}, \frac{g^{-w t_2} \cdot h_2^{-r_2 t_2}}{g^{-w t_2} \cdot h_0^{-r_0 t_2}}) &= \mathbf{e}(\frac{g^{r_0 t_1 t_2}}{g^{r_2 t_1 t_2}}, \frac{g^{-w t_2} \cdot h_0^{-r_0 t_2}}{g^{-w t_2} \cdot h_1^{-r_1 t_2}}) \Leftrightarrow \\
\mathbf{e}(\frac{g^{r_1 t_1 t_2}}{g^{r_0 t_1 t_2}}, \frac{h_2^{-r_2 t_2}}{h_0^{-r_0 t_2}}) &= \mathbf{e}(\frac{g^{r_0 t_1 t_2}}{g^{r_2 t_1 t_2}}, \frac{h_0^{-r_0 t_2}}{h_0^{-r_1 t_2}}) \Leftrightarrow \\
\mathbf{e}(g^{(r_1 - r_0)}, h_0^{r_0} \cdot h_2^{-r_2})^{t_1 (t_2)^2} &= \mathbf{e}(g^{(r_0 - r_2)}, h_0^{(r_1 - r_0)})^{t_1 (t_2)^2} \Leftrightarrow \\
\mathbf{e}(g, h_0^{r_0} \cdot h_2^{-r_2}) &= \mathbf{e}(g, h_0^{(r_0 - r_2)}) \Leftrightarrow \\
h_2 &= h_0
\end{aligned}$$

Therefore, we come to the interesting conclusion that AIBE_1 is PRIV-TO secure against $(2, 0)$-bounded samplers, yet completely insecure against $(t, 0)$-bounded samplers, for $t > 2$.

6.1.2 An extended version over composite-order groups

We extend AIBE_1 to groups of composite order and obtain AIBE_2 [Figure 6.5]. The extension is very simple: let all the parameters in the original scheme be from the subgroup $\mathbb{G}_p$ (generated by g_p) and randomize each element of the extracted secret key by a random element from the subgroup $\mathbb{G}_q$ (generated by g_q). Note that the message space is $\mathbb{G}_T$.

```
$\underline{\mathsf{AIBE.Setup}(1^\lambda)}$:
$(p, q, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g) \leftarrow\$ \mathcal{G}_\mathcal{C}(1^\lambda)$
$n \leftarrow pq;\quad g_p \leftarrow g^q;\quad g_q \leftarrow g^p$
$\Gamma \leftarrow (n, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, g_p, g_q)$
$w, t_1, t_2 \leftarrow\$ \mathbb{Z}_n$
$\Omega \leftarrow \mathbf{e}(g_p, g_p)^{t_1 t_2 w}$
$v_1 \leftarrow g_p^{t_1}$
$v_2 \leftarrow g_p^{t_2}$
$h \leftarrow\$ \mathcal{H}_\lambda$
$mpk \leftarrow (\Gamma, \Omega, v_1, v_2, h)$
$msk \leftarrow (w, t_1, t_2)$
return $(msk, mpk)$
```

```
$\underline{\mathsf{AIBE.KeyGen}(mpk, msk, id)}$:
$(w, t_1, t_2) \leftarrow msk$
$(\Gamma, \Omega, v_1, v_2, h) \leftarrow mpk$
$(n, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, g_p, g_q) \leftarrow \Gamma$
$r \leftarrow\$ \mathbb{Z}_n$
$x_0, x_1, x_2 \leftarrow\$ \mathbb{G}_q$
$h \leftarrow h(id)$
$d_0 \leftarrow x_0 \cdot g_p^{r t_1 t_2}$
$d_1 \leftarrow x_1 \cdot g_p^{-w t_2} \cdot h^{-r t_2}$
$d_2 \leftarrow x_2 \cdot g_p^{-w t_1} \cdot h^{-r t_1}$
$sk \leftarrow (d_0, d_1, d_2)$
return $sk$
```

```
$\underline{\mathsf{AIBE.Enc}(mpk, m, id)}$:
$(\Gamma, \Omega, v_1, v_2, h) \leftarrow mpk$
$(n, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, g_p, g_q) \leftarrow \Gamma$
$s, s_1 \leftarrow\$ \mathbb{Z}_n$
$h \leftarrow h(id)$
$\hat{c} \leftarrow \Omega^s m$
$c_0 \leftarrow h^s$
$c_1 \leftarrow v_1^{s - s_1}$
$c_2 \leftarrow v_2^{s_1}$
$c \leftarrow (\hat{c}, c_0, c_1, c_2)$
return $c$
```

```
$\underline{\mathsf{AIBE.Dec}(mpk, sk_{id}, c)}$:
$(\Gamma, \Omega, v_1, v_2, h) \leftarrow mpk$
$(n, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, g_p, g_q) \leftarrow \Gamma$
$(d_0, d_1, d_2) \leftarrow sk_{id}$
$(\hat{c}, c_0, c_1, c_2) \leftarrow c$
$e_0 \leftarrow \mathbf{e}(c_0, d_0)$
$e_1 \leftarrow \mathbf{e}(c_1, d_1)$
$e_2 \leftarrow \mathbf{e}(c_2, d_2)$
$m \leftarrow \hat{c} \cdot e_0 \cdot e_1 \cdot e_2$
return $m$
```

Figure 6.5: AIBE_2, an extended version of AIBE scheme AIBE_1, over composite-order groups.

The decryption algorithm remains correct, since

$$
\begin{aligned}
e_0 &= \mathbf{e}(h^s, x_0 \cdot g_p^{rt_1t_2}) = \mathbf{e}(h^s, g_p^{rt_1t_2})\\
e_1 &= \mathbf{e}(v_1^{s-s_1}, x_1 \cdot g_p^{-wt_2} \cdot h^{-rt_2}) = \mathbf{e}(v_1^{s-s_1}, g_p^{-wt_2} \cdot h^{-rt_2})\\
e_2 &= \mathbf{e}(v_2^{s_1}, x_2 \cdot g_p^{-wt_1} \cdot h^{-rt_1}) = \mathbf{e}(v_2^{s_1}, g_p^{-wt_1} \cdot h^{-rt_1})
\end{aligned}
$$

Also, IND-CPA secure is preserved, assuming DBDH and DLIN hold in $\mathbb{G}_p$. However, we show that the extra randomization of the decryption keys enhances the function privacy guarantees (previously limited to only two keys) to an arbitrary number.

First, we introduce a new hardness assumption over composite order groups. We call this assumption the *Composite Decisional Diffie-Hellman* (CDDH) and show that it is weaker than the *Composite 3-party Diffie-Hellman* (C3DH) assumption made in [BW07] by Boneh and Waters.

Definition 4. *We say the CDDH assumption holds for group generator $\mathcal{G}_{\mathcal{C}}$ if for every ppt adversary $\mathcal{A}$ we have that*

$$\mathbf{Adv}^{\mathsf{cddh}}_{\mathcal{G}_{\mathcal{C}},\mathcal{A}}(\lambda) := 2 \cdot \Pr[\mathsf{CDDH}^{\mathcal{A}}_{\mathcal{G}_{\mathcal{C}}}(1^\lambda)] - 1 \in \textsc{Negl},$$

where game CDDH *is described in Figure 6.6.*

```
$\underline{\mathrm{CDDH}^{\mathcal{A}}_{\mathcal{G}_{\mathcal{C}}}(1^\lambda)}$:
$(p, q, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g) \leftarrow_\$ \mathcal{G}_{\mathcal{C}}(1^\lambda)$
$n \leftarrow pq$;   $g_p \leftarrow g^q$;   $g_q \leftarrow g^p$
$\Gamma \leftarrow (n, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, g_p, g_q)$
$X_1, X_2, X_3 \leftarrow_\$ \mathbb{G}_q$
$a, b \leftarrow_\$ \mathbb{Z}_n$
$b \leftarrow_\$ \{0, 1\}$
if $(b = 0)$ then $R \leftarrow X_3(g_p)^{ab}$
else $R \leftarrow_\$ \mathbb{G}$
$b' \leftarrow_\$ \mathcal{A}(\Gamma, X_1(g_p)^a, X_2(g_p)^b, R)$
return $(b = b')$
```

Figure 6.6: Game defining CDDH computational assumption.

In game C3DH, adversary is given a tuple $(\Gamma', g_p, g_q, (g_p)^a, (g_p)^b, X_1(g_p)^{ab}, X_2(g_p)^{abc}, Z)$ and has to decide whether $Z = X_3(g_p)^c$, for some $X_3 \in \mathbb{G}_q$. For convenience, we rewrite this as $(\Gamma', \mathsf{g_p}, \mathsf{g_q}, (\mathsf{g_p})^{\mathsf{a}}, (\mathsf{g_p})^{\mathsf{b}}, \mathsf{X}_1(\mathsf{g_p})^{\mathsf{ab}}, \mathsf{Y}, \mathsf{X}_3(\mathsf{g_p})^{\mathsf{c}})$, where Y is either $\mathsf{X}_2(\mathsf{g_p})^{\mathsf{abc}}$ or random in $\mathbb{G}$. Now, notice that $(\Gamma', \mathsf{g_p}, \mathsf{g_q}, \mathsf{X}_1(\mathsf{g_p})^{\mathsf{ab}}, \mathsf{X}_3(\mathsf{g_p})^{\mathsf{c}}, \mathsf{Y})$ is a CDDH tuple. Therefore, CDDH is a weaker assumption than C3DH.

We now show our main result for this construction. Assuming the hardness of CDDH, we prove that AIBE_2 is PRIV-TO secure for an arbitrary number of challenge keys.

Theorem 4. *The anonymous identity-based encryption scheme* AIBE_2 *[Figure 6.5] is* PRIV-TO *secure against unpredictable samplers that are functionally-differing, in the random oracle model, assuming CDDH is intractable [Definition 4].*

Proof. Let $(\mathcal{A}, \mathcal{S})$ be a legitimate adversary against the PRIV-TO security of AIBE_2, where $\mathcal{S}$ is functionally-differing and outputs two vectors of t identities. The proof follows a hybrid argument. The challenger starts by extracting decryption keys from the left side, as if $b = 0$. Then, through a sequence of game, the challenger extracts the keys from random identities, replacing one key at the time. We argue the indistinguishability of each game hop with the hardness of CDDH. Finally, the challenger proceeds extracting each key from the right vector of identities output by the sampler, again, replacing one challenge key at the time. At the end, the challenger extracts decryption keys from the right side, as if $b = 1$.

Game$_0$: This game is identical to PRIV-TO game when the challenge bit $b = 0$.

Game$_1$: In this game, we set a bad event in case the adversary $\mathcal{A}$ queries to the random oracle any of the identities output by $\mathcal{S}$. This hop is down to the unpredictability of the sampler, given that $\mathcal{S}$ is functionally-differing.

Game$_{(2,i)}$, for $i \in \{0, t\}$: Instead of extracting a decryption key for $\mathbf{id}_0[i]$, we extract a decryption key for a random $\mathsf{id} \in \mathsf{IdSp}_\lambda$. Game$_{(2,0)}$ is the same as Game$_1$. In Game$_{(2,t)}$ all decryption keys of the challenge are extracted from independently sampled random identities. We construct an adversary $\mathcal{B}_i$ that plays the CDDH game and simulates either Game$_{(2,(i-1))}$ or Game$_{(2,i)}$, depending on the random bit of game CDDH. $\mathcal{B}_i$ sets $h(\mathbf{id}_0[i])$ to $(\mathsf{g_p})^{\mathsf{a}}$, and to extract sk_i it sets the randomness r to b. Notice that sk_i is easily computable with the CDDH tuple, even without actually knowing the value of $(\mathsf{g_p})^{\mathsf{a}}$ or b. If $\mathcal{B}_i$ has to extracts several keys for

$h(\mathbf{id}_0[i])$—because of the sampler's choices—it extracts all the other decryption keys with independent random coins, instead of fixing r to be b. In this way, sk_i is a decryption key for identity $\mathbf{id}_0[i]$ if the CDDH tuple is well-formed, and is a decryption key for a uniformly sampled identity if it is not. Thus, the distance between $\mathrm{Game}_{(2,(i-1))}$ and $\mathrm{Game}_{(2,i)}$ is bounded by the advantage of $\mathcal{B}_i$ against game CDDH. We describe the internal functioning of algorithm $\mathcal{B}_i$ in Figure 6.7.

$\mathrm{Game}_{(3,i)}$**, for** $i \in \{0, t\}$: Instead of extracting the ith decryption from a random $\mathsf{id} \in \mathsf{IdSp}_\lambda$, we extract the decryption key from $\mathbf{id}_1[i]$. The argument is similar to the previous game hops.

Game_4 : In this game, we remove the bad event introduced earlier in Game_1. Again, this hop is down to the unpredictability of $\mathcal{S}$. This game is identical to PRIV-TO game when the challenge bit $b = 1$.

Therefore, we have that

$$\mathbf{Adv}^{\text{priv-to}}_{\mathsf{AIBE}_2,(\mathcal{S},\mathcal{A})}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\text{pred}}_{\mathcal{S},\mathcal{P}}(\lambda) + 2t \cdot \mathbf{Adv}^{\mathsf{cddh}}_{\mathcal{G}_\mathcal{C},\mathcal{B}}(\lambda).$$

□

```
$\underline{\mathcal{B}_i(\Gamma, \mathsf{Z_a}, \mathsf{Z_b}, \mathsf{Z_{ab}})}$:
$(\mathsf{n}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}, \mathsf{g_p}, \mathsf{g_q}) \leftarrow \Gamma$
$\mathsf{w}, \mathsf{t}_1, \mathsf{t}_2 \leftarrow\$ \mathbb{Z}_\mathsf{n}$
$\Omega \leftarrow \mathbf{e}(\mathsf{g_p}, \mathsf{g_p})^{\mathsf{t_1 t_2 w}}$
$\mathsf{v}_1 \leftarrow \mathsf{g_p^{t_1}}$
$\mathsf{v}_2 \leftarrow \mathsf{g_p^{t_2}}$
$\mathsf{mpk} \leftarrow (\Gamma, \Omega, \mathsf{v}_1, \mathsf{v}_2)$
$\mathsf{msk} \leftarrow (\mathsf{w}, \mathsf{t}_1, \mathsf{t}_2)$
$b \leftarrow\$ \{0, 1\}$
$b' \leftarrow\$ \mathcal{A}^{\mathrm{H,TGEN,LR}}(\mathsf{mpk})$
return $(b = b')$

$\underline{\mathrm{H}(\mathsf{id})}$:
$\mathsf{h} \leftarrow \mathsf{list}[\mathsf{id}]$
if $(\mathsf{h} = \perp)$
  $\mathsf{h} \leftarrow\$ \mathbb{G}_\mathsf{p}$
  $\mathsf{list}[\mathsf{id}] \leftarrow \mathsf{h}$
return $\mathsf{h}$

$\underline{\mathrm{TGEN}(\mathsf{id})}$:
$\mathsf{sk_{id}} \leftarrow\$ \mathsf{AIBE.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$
return $\mathsf{sk_{id}}$

$\underline{\mathrm{LR}(st)}$:
$(\mathbf{id}_0, \mathbf{id}_1) \leftarrow\$ \mathcal{S}(st)$
for $j \in \{0, (i-1)\}$
  $\mathsf{id} \leftarrow\$ \mathsf{IdSp}_\lambda$
  $\mathbf{sk}[j] \leftarrow\$ \mathsf{AIBE.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$
$\mathsf{id}^\star \leftarrow \mathbf{id}_0[i]$
$\mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_2 \leftarrow\$ \mathbb{G}_\mathsf{q}$
$\mathsf{d}_0 \leftarrow \mathsf{x}_0 \cdot \mathsf{Z_b^{t_1 t_2}}$
$\mathsf{d}_1 \leftarrow \mathsf{x}_1 \cdot \mathsf{Z_{ab}^{-t_2}} \cdot (\mathsf{g_p})^{\mathsf{-wt_2}}$
$\mathsf{d}_2 \leftarrow \mathsf{x}_2 \cdot \mathsf{Z_{ab}^{-t_1}} \cdot (\mathsf{g_p})^{\mathsf{-wt_1}}$
$\mathbf{sk}[i] \leftarrow (\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2)$
for $j \in \{(i+1), t\}$
  if $(\mathbf{id}_0[j] = \mathsf{id}^\star)$
    $\mathsf{r} \leftarrow\$ \mathbb{Z}_\mathsf{n}$
    $\mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_2 \leftarrow\$ \mathbb{G}_\mathsf{q}$
    $\mathsf{d}_0 \leftarrow \mathsf{x}_0 \cdot \mathsf{g_p^{rt_1 t_2}}$
    $\mathsf{d}_1 \leftarrow \mathsf{x}_1 \cdot \mathsf{Z_a^{-rt_2}} \cdot (\mathsf{g_p})^{\mathsf{-wt_2}}$
    $\mathsf{d}_2 \leftarrow \mathsf{x}_2 \cdot \mathsf{Z_a^{-rt_1}} \cdot (\mathsf{g_p})^{\mathsf{-wt_1}}$
    $\mathbf{sk}[j] \leftarrow (\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2)$
  else
    $\mathbf{sk}[j] \leftarrow\$ \mathsf{AIBE.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{id}_0[j])$
return $\mathbf{sk}$
```

Figure 6.7: CDDH adversary $\mathcal{B}_i$, as part of proof of Theorem 4.

6.2 The obfuscate-extract (OX) transform

Our first black-box construction formalizes the intuition that obfuscating circuits before computing a token for them will provide some form of token privacy.

THE OX TRANSFORM. Let Obf be an obfuscator supporting a circuit family CSp and let FE be a functional encryption scheme supporting all polynomial-size circuits. We construct a functional encryption scheme OX[FE, Obf] via the OX transform as follows. Setup, encryption and evaluation algorithms are identical to those of the base functional encryption scheme. The token-generation algorithm creates a token for the circuit that results from obfuscating the extracted circuit, i.e. $\mathsf{OX[FE, Obf].TGen(msk, C)} := \mathsf{FE.TGen(msk, Obf}(1^\lambda, \mathsf{C}))$. Correctness of this construction follows from those of its underlying components. We now show that this construction yields function privacy against PRIV-TO adversaries. Since PRIV-TO does not imply IND-CPA security—see the discussion in Section 5.3—we establish IND-CPA security independently. The proof of the following theorem is straightforward and results from direct reductions to the base FE and Obf schemes used in the construction.

Theorem 5 (OX is PRIV-TO $\wedge$ IND-CPA)**.** *If obfuscator* Obf *is* DI *secure, then scheme* $\mathsf{OX}[\mathsf{FE}, \mathsf{Obf}]$ *is* PRIV-TO *secure. Furthermore, if* FE *is* IND-CPA *secure* $\mathsf{OX}[\mathsf{FE}, \mathsf{Obf}]$ *is* IND-CPA *secure.*

Proof. The proof is straightforward and results from direct reductions to the underlying components used in the construction. We start by proving that $\mathsf{OX}[\mathsf{FE}, \mathsf{Obf}]$ is PRIV-TO-secure for a circuits family CSp and (circuits-only) sampler class $\mathbb{S}$ if Obf is DI-secure for CSp and $\mathbb{S}$. Given an adversary $(\mathcal{S}, \mathcal{A}_1)$ against PRIV-TO security of $\mathsf{OX}[\mathsf{FE}, \mathsf{Obf}]$, we construct an adversary $(\mathcal{S}', \mathcal{B}_1)$ against the DI security of Obf as follows. We set $\mathcal{S}'$ to be the same as $\mathcal{S}$. Algorithm $\mathcal{B}_1$ runs $\mathsf{FE.Gen}(1^\lambda)$ to generate on its own a master secret key and master public key pair $(\mathsf{msk}, \mathsf{mpk})$. Then, $\mathcal{B}_1$ runs $\mathcal{A}_1$ on mpk, answering all its token-generation queries by running $\mathsf{FE.TGen}(\mathsf{msk}, \cdot)$, until $\mathcal{A}_1$ calls LR on some state *st*. At this point, $\mathcal{B}_1$ calls its own LR oracle on *st* and receives as a challenge a vector of obfuscated circuits. $\mathcal{B}_1$ generates a token for each circuit, and forwards the result to $\mathcal{A}_1$. Thereafter, $\mathcal{B}_1$ continues running $\mathcal{A}_1$, answering its second-stage token-generation queries as before until $\mathcal{A}_1$ outputs a bit b', which $\mathcal{B}_1$ outputs as its own guess. The simulation is perfect and $\mathcal{S}'$ is unpredictable because $\mathcal{S}$ is unpredictable.

We now prove the second part of the theorem, i.e. that $\mathsf{OX}[\mathsf{FE}, \mathsf{Obf}]$ is IND-CPA-secure if FE is. Let $\mathcal{A}_2$ be an adversary against IND-CPA security of $\mathsf{OX}[\mathsf{FE}, \mathsf{Obf}]$. We construct an adversary $\mathcal{B}_2$ against IND-CPA security of FE. Algorithm $\mathcal{B}_2(\mathsf{mpk})$ runs $\mathcal{A}_2(\mathsf{mpk})$, answering its first-stage $\textsc{TGen}(\mathsf{C})$ queries by first computing an obfuscation $\overline{\mathsf{C}}$ of circuit C, placing a token-generation query on $\overline{\mathsf{C}}$ to its own $\textsc{TGen}$ oracle, and forwarding the token to $\mathcal{A}_2$. When $\mathcal{A}_2$ asks to be challenged on messages $(\mathsf{m}_0, \mathsf{m}_1)$, $\mathcal{B}_2$ calls its own LR oracle on these messages and forwards the challenge ciphertext to $\mathcal{A}_2$. Second-stage $\textsc{TGen}$ queries are answered as before. Finally, $\mathcal{B}_2$ outputs $\mathcal{A}_2$'s guess b' as its own guess. Here again, the simulation is perfect and legitimacy of $\mathcal{B}_2$ follows from the legitimacy of $\mathcal{A}_2$ and the fact that the obfuscator preserves the functionality of the circuit, which means that $\mathcal{B}_2$ has precisely the same advantage as $\mathcal{A}_2$. Therefore, we conclude that

$$\mathbf{Adv}^{\text{priv-to}}_{\mathsf{OX[FE,Obf]},\mathcal{S},\mathcal{A}_1}(\lambda) = \mathbf{Adv}^{\text{di}}_{\mathsf{Obf},\mathcal{S},\mathcal{B}_1}(\lambda), \text{ and that}$$
$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{OX[FE,Obf]},\mathcal{A}_2}(\lambda) = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{FE},\mathcal{B}_2}(\lambda) \ .$$

□

We note that this proof holds for arbitrary classes of circuits and arbitrary (circuits-only) samplers. Using the composable VGB point-function obfuscator of Bitansky and Canetti [BC14] and any secure functional encryption scheme that is powerful enough to support one exponentiation and one equality test (e.g. supports $\mathbf{NC}^1$ circuits) we obtain a private keyword search scheme in the presence of tokens for arbitrarily correlated keywords. If the underlying functional encryption scheme supports the more powerful functionality that permits attaching a payload to the point, one obtains a PRIV-TO anonymous identity-based encryption scheme where arbitrary correlations are tolerated. In this case, on input $(\mathsf{id}, \mathsf{m})$, the functionality supported by the underlying FE scheme would return m if $\overline{\mathsf{C}}(\mathsf{id}) = 1$, where $\overline{\mathsf{C}}$ was sampled from $\mathsf{Obf}(\mathsf{C}[\mathsf{id}^\star])$ during token generation; it would return $\perp$ otherwise.

The above theorem shows that DI is sufficient to build a PRIV-TO scheme. It is however easy to see that the existence of a single-circuit DI obfuscator is also necessary. Indeed, given any PRIV-TO scheme FE we can DI-obfuscate a single circuit C by generating a fresh FE key pair, and outputting $\mathsf{FE.Eval}(\cdot, \mathsf{tk})$ where tk is a token for C.

Proposition 4 (PRIV-TO vs. DI)**.** *A* PRIV-TO*-secure functional encryption for a circuits family* CSp *exists if a* DI *obfuscator for* CSp *exists. Conversely, a single-circuit* DI *obfuscator for* CSp *exists if a* PRIV-TO*-secure functional encryption for* CSp *exists.*

Proof. We first describe the operation of the required obfuscator. Given a circuit C, the required obfuscator Obf generates an FE key pair $(\mathsf{mpk}, \mathsf{msk})$ and uses the master secret key to extract a token tk for C. It then defines the obfuscated circuit to be one that first encrypts m under mpk using trivial random coins, and then evaluates the resulting ciphertext using tk, i.e. the circuit

$$\overline{\mathsf{C}}[\mathsf{mpk}, \mathsf{tk}](\cdot) := \mathsf{FE.Eval}(\mathsf{FE.Enc}(\cdot, \mathsf{mpk}; 0^{\mathsf{poly}(\lambda)}), \mathsf{tk}) \ .$$

The correctness of this obfuscator follows from that of the FE scheme. The proof of DI security for this construction with respect to samplers that output a single circuit pair is a direct reduction to PRIV-TO. We construct a PRIV-TO adversary that uses the DI sampler without change and does nothing in the first stage. In the second stage, on obtaining the challenge token tk, constructs $\overline{\mathsf{C}}[\mathsf{mpk}, \mathsf{tk}](\cdot)$ and passes it on to the DI distinguisher. It will return whatever the distinguisher outputs. This simulation is easily seen to be perfect. □

Remark. For arbitrary DI samplers the argument above fails. This is due to the fact that communication between the sampler and the distinguisher is restricted (by the unpredictability condition) and hence hybrid arguments cannot be made to go through.

A similar line of reasoning shows that the extractor-based constructions of private FE by Boneh, Raghunathan and Segev [BRS13a] and Arriaga, Tang and Ryan [ATR14] give rise to single-circuit DI obfuscators for point functions for the specific classes of samplers considered in those works.

Agrawal et al. [AAB+15] have proposed a simulation-based definition of privacy that strikes a different balance between practical relevance and feasibility. However, the definition in [AAB+15] implies VBB obfuscation, which is known to be feasible only for restricted classes of circuits [BR14a, BBC+14] or in idealized models of computation [CV13, BR14b, BGK+14]. The above proposition shows that our model is closer to the weaker form of DI obfuscation, which as shown in Proposition 2 is implied by CVGB. The particular case of single-circuit DI obfuscation is implied by VGB (and hence VBB) obfuscation. Therefore, our model is more amenable to instantiations in the standard model.

6.3 The trojan-obfuscate-extract (TOX) transform

We now present a generic construction that achieves Res-PRIV security for a class of samplers that we call *concentrated.* To this end, we build on the ideas from [ABSV15, DIJ+13] on converting selective to adaptive security and achieving simulation-based security from IND-CPA security for FE schemes.

THE TOX TRANSFORM. Given a symmetric encryption scheme SE, a general-purpose obfuscator Obf and a functional encryption FE for all circuits, our *Trojan-Obfuscate-Extract* (TOX) transform operates as follows. The master public key of the scheme is the same as that of the base FE scheme. Its master secret key includes a symmetric key k and the master secret key for the base FE scheme. To encrypt a message m we call the base FE encryption routine on $(0, 0^\lambda, \mathsf{m})$. To generate a token for a circuit C, we first generate an obfuscation $\bar{\mathsf{C}} \leftarrow_\$ \mathsf{Obf}(\mathsf{C})$, a ciphertext $\mathsf{c} \leftarrow_\$ \mathsf{SE.Enc}(\mathsf{k}, 0^n)$ and construct the following circuit.

$$\mathsf{Troj}[\bar{\mathsf{C}}, \mathsf{c}](\mathsf{b}, \mathsf{k}, \mathsf{m}) := \begin{cases} \bar{\mathsf{C}}(\mathsf{m}) & \text{if } \mathsf{b} = 0 \; ; \\ \mathsf{C}^*(\mathsf{m}) & \text{if } \mathsf{b} = 1, \text{ where } \mathsf{C}^* = \mathsf{SE.Dec}(\mathsf{k}, \mathsf{c}) \; . \end{cases}$$

Finally, we extract a token for $\mathsf{Troj}[\bar{\mathsf{C}}, \mathsf{c}]$. Evaluation simply invokes the corresponding operation in the underlying FE.

The correctness and IND-CPA security of this construction follow easily from the correctness and IND-CPA security of the underlying functional encryption scheme via straightforward reductions. Intuitively, during the normal operation of the scheme, the tokens in the construction will simply evaluate an obfuscation of the extracted circuit. In the proof of privacy, however, we will take advantage of the fact that a totally independent circuit can be hidden inside the token within the symmetric encryption ciphertext, and unlocked by a message containing the correct symmetric decryption key. For the proof to go through, the hidden circuit must be carefully selected so that the legitimacy condition is observed throughout. In order to meet this latter restriction, we consider the following constrained class of samplers.

CONCENTRATED SAMPLERS. We say a sampler $\mathcal{S}$ is $\mathcal{S}^*$-concentrated if for all st, all CSp_λ-vectors $\mathbf{C}$ we have that

$$\Pr\left[\mathbf{C}(\mathbf{m}_0) = \mathbf{C}(\mathbf{m}_1) \neq \mathbf{C}(\mathbf{m}^*)\right] \in \text{NEGL} \text{ and } \Pr\left[\mathbf{C}_0(\mathbf{m}_0) \neq \mathbf{C}^*(\mathbf{m}^*)\right] \in \text{NEGL},$$

where the probability space of these is defined by the operations $(\mathbf{C}^*, \mathbf{m}^*) \leftarrow_\$ \mathcal{S}^*(z, \mathbf{C})$ and $(\mathbf{C}_0, \mathbf{C}_1, \mathbf{m}_0, \mathbf{m}_1, z) \leftarrow_\$ \mathcal{S}(st)$.

Concentration is a property independent of unpredictability and we will be relying on both in our construction. Unpredictability is used in the reduction to the DI assumption.

Concentration guarantees the existence of a sampler $\mathcal{S}^*$ that generates circuits $\mathbf{C}^*$ and messages $\mathbf{m}^*$ which permit decoupling circuits and messages in the security proof. Intuitively, quantification over all $\mathbf{C}$ means that adversarially generated circuits will lead to image matrices that collide with those leaked by the sampler with overwhelming probability. The additional restriction on $\mathbf{C}^*(\mathbf{m}^*)$ guarantees that one can switch from the honest branch of challenge tokens to one corresponding to the trojan branch. Both of these properties are important to guarantee legitimacy when making a reduction to the security of the FE scheme. We however need to impose that legitimacy also holds for second-phase TGen queries as well, and this is where we need to assume Res-PRIV security: the extra legitimacy condition allows us to ensure that by moving to $\mathbf{m}^*$ the legitimacy condition is not affected in the second phase either. Finally, an important observation is that, because we are dealing with concentrated samplers, our security proof goes through assuming obfuscators that need only tolerate random auxiliary information.

Theorem 6 (TOX is Res-PRIV)**.** *If obfuscator* Obf *is* DI *secure,* SE *is* IND-CPA *secure and* FE *is* IND-CPA *secure, then scheme* $\mathsf{TOX}[\mathsf{FE}, \mathsf{Obf}, \mathsf{SE}]$ *is* Res-PRIV *secure with respect to concentrated samplers.*

Proof. The proof proceeds via a sequence of three games as follows.

Game$_0$: This game is identical to Res-PRIV: challenge vector $\mathbf{C}_b$ is extracted and $\mathbf{m}_b$ is encrypted for a random bit b and for all TGen queries, string 0^n is encrypted using SE in the trojan branch.

Game$_1$: In this game, instead of 0^n we encrypt the circuits queried to the (first or second-phase) TGen oracle under a symmetric key k^* in the trojan branch. In the challenge phase, we sample $(\mathbf{C}^*, \mathbf{m}^*) \leftarrow_\$ \mathcal{S}^*(z, \mathbf{C})$, where $\mathbf{C}$ are all first-phase TGen queries, and encrypt $\mathbf{C}^*$ under k^* for the challenge circuits in the trojan branch. This transition is negligible down to IND-CPA security of SE.

Game$_2$: In this game, instead of encrypting $(0, 0, \mathbf{m}_b)$ we encrypt $(1, \mathsf{k}^*, \mathbf{m}^*)$ in the challenge phase where the latter is generated using $\mathcal{S}^*(z, \mathbf{C})$. We reduce this hop to the IND-CPA security of FE. We generate a key k^*, answer first-stage TGen queries using the provided TGen oracle and encrypt circuits under k^* in the trojan branch to get st. We run $\mathcal{S}(st)$ and get $(\mathbf{C}_0, \mathbf{m}_0, \mathbf{C}_1, \mathbf{m}_1, z)$. We then run $\mathcal{S}^*(z, \mathbf{C})$, where $\mathbf{C}$ are all first-phase TGen queries, to get $(\mathbf{C}^*, \mathbf{m}^*)$. We prepare challenges tokens by encrypting $\mathbf{C}^*$ under k^* in the trojan branch and using the provided TGen oracle we generate the challenge tokens. We query the provided LR on $(0, 0, \mathbf{m}_b)$ and $(1, \mathsf{k}^*, \mathbf{m}^*)$ and receive the corresponding vector of ciphertexts. Second-stage TGen queries are handled using provided TGen oracle and k^*. Finally, we return the same bit that the distinguisher returns. Legitimacy of first-stage TGen queries follows from the first condition on concentration that with high probability $\mathbf{C}(\mathbf{m}_b) = \mathbf{C}(\mathbf{m}^*)$. For the challenge tokens, this follows from the second concentration requirement that $\mathbf{C}_b(\mathbf{m}_b) = \mathbf{C}^*(\mathbf{m}^*)$. For the second-stage queries we rely on the restriction on the adversary. Recall that in the Res-PRIV model, any second-stage queries must have an image vector which matches one for a first-stage query. Since the first-stage images match those on $\mathbf{m}^*$ (and hence are legitimate), the second-stage ones will be also legitimate. We output $(b' = b)$ where the distinguisher outputs b'. As a result of this game, the challenge messages no longer depend on b. It is easy to see that according to the IND-CPA challenge bit this reduction interpolates between games Game_1 and Game_2.

Game_3 : In this game we use $\mathbf{C}_1$ in challenge token generation even if $b = 0$. We show this hop in unnoticeable down to the security of the obfuscator. We sample an FE key pair and a symmetric key and simulating the first-stage TGen queries for the adversary as before. We define a DI sampler that outputs the circuits that the Res-PRIV sampler outputs, but extends the circuit list to include another copy of $\mathbf{C}_1$ on both sides. This sampler also outputs as auxiliary information z' the original auxiliary information output by the PRIV sampler, extended with the random coins used to generate the FE key, the symmetric key, and to run the first stage of the adversary (this will allow the second stage DI adversary to reconstruct the keys and first stage TGen queries). It follows that this sampler is unpredictable as long as the Res-PRIV sampler is. When we receive the obfuscations and z', we generate $(\mathbf{C}^*, \mathbf{m}^*) \leftarrow_\$ \mathcal{S}^*(z, \mathbf{C})$, where $\mathbf{C}$ are all first-phase TGen queries. We form the challenge tokens using the received obfuscations and $\mathbf{C}^*$, taking the $\mathbf{C}_1$ obfuscations from the duplicated part of the challenge, and the $\mathbf{C}_0$ obfuscations from the original part (these can now be either $\mathbf{C}_0$ or $\mathbf{C}_1$ depending on the external challenge bit). Challenge ciphertexts are generated by encrypting $\mathbf{m}^*$ (rules of Game$_2$). We answer the second-stage TGen queries using the FE key and the symmetric key. We return whatever the distinguisher returns. It is easy to see that according to the DI challenge bit this reduction interpolates between games Game$_2$ and Game$_3$.

In Game$_3$ both the challenge tokens and challenge ciphertexts are independent of the bit b and hence the advantage of any adversary is 0. □

Examples. Consider keyword samplers which output high-entropy keywords and messages with arbitrary image matrices. All such samplers are concentrated around a sampler $\mathcal{S}^*$ that outputs uniformly random keywords and messages subject to the same image pattern. The second concentration condition is immediate and the first follows from the fact that all messages and circuits have high entropy and $\mathbf{C}$ is selectively chosen. Although this argument can be extended to samplers outputting low-entropy keywords whose complete image matrix is predictable or is included in z, the latter requirement may not always be the case in general. Consider, for example, a vector $\mathbf{C}$ consisting of circuits for $\mathsf{w} = 0^n$ and messages $\mathbf{m}_0 = \mathbf{m}_1$ whose components are randomly set to 0^n and 1^n. The image matrix in this setting is unpredictable as long as a sufficiently large number of messages are output.

Hyperplane membership circuits $\mathsf{C}[\mathbf{v}](\mathbf{w})$ return 1 iff $\langle \mathbf{v}, \mathbf{w} \rangle = 0 \pmod p$ for a prime p. As another example, consider unpredictable samplers of hyperplane membership circuits that output n vectors $\mathbf{v}_i \in \mathbb{Z}_p^d$ and m messages $\mathbf{w}_i \in \mathbb{Z}_p^d$ where all vector entries have high entropy. Given the corresponding $n \times m$ image matrix, whenever $d(n+m) > nm$, a high-entropy pre-image to the image matrix can be sampled as the system will be underdetermined. Under this condition, the second requirement needed for concentration is met, and the first condition follows as this pre-image is high entropy and $\mathbf{C}$ is selectively chosen. This observation implies that a DI obfuscator for the hyperplane membership problem will immediately yield a private functional encryption scheme for the same functionality under arbitrary correlations via the TOX construction, a problem that was left open in [BRS13b]. In Section 4.3.2 we gave a direct construction of a DI obfuscator for hyperplane membership by proving that the obfuscator of Canetti, Rothblum and Varia [CRV10] is DI secure in the presence of random auxiliary information under a variant of the DDH assumption in the style of those used in [CRV10, BC14].

6.4 The disjunctively-obfuscate-extract (DOX) transform

In this section, we present a construction specific to point functions. We were able to remove the limitation of the TOX transform that provides security guarantees only against concentrated samplers, and achieve privacy in the presence of arbitrary correlations between searched keywords and encrypted messages. Our construction demands less from the underlying functional encryption and obfuscator, and hence can potentially allow more efficient instantiations of these primitives.

The DOX transform. Let Obf be an obfuscator supporting a point circuit family CSp over message space MSp. Let FE be a functional encryption scheme supporting general circuits, and let PRP be a pseudorandom permutation (see Section 2.3.1 for the formal definition). We construct a keyword search scheme KS for keyword space $\mathsf{WSp} = \mathsf{MSp}$ via the *Disjunctively-Obfuscate-Extract* (DOX) transform as follows. The key-generation algorithm samples a PRP key $\mathsf{k} \leftarrow_{\$} \mathsf{K}(1^\lambda)$ and an FE key pair $(\mathsf{msk}, \mathsf{mpk}) \leftarrow_{\$} \mathsf{FE.Gen}(1^\lambda)$. It returns $((\mathsf{k}, \mathsf{msk}), \mathsf{mpk})$. The encryption operation is identical to that of the FE scheme. The test algorithm is identical to the evaluation algorithm of FE. The token-generation algorithm computes

$$\mathsf{FE.TGen}(\mathsf{msk}, \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}]) \vee \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{E}(\mathsf{k}, \mathsf{w})])) \ .$$

The FE-extracted circuits are two-point circuits implemented as the disjunction of two obfuscated point functions. One of the points will correspond to the searched query, whereas the other point will be pseudorandom and will be only used for proofs of security. (In a loose sense, the second point represents the second branch in the TOX construction.)

As in OX, the composable VGB obfuscator of [BC14] for point functions and any general-purpose functional encryption scheme (such as those in [GKP+13, GGH+13]) can be used to instantiate the above construction. The supported circuit class would roughly amount to two parallel group operations and two comparisons in a DDH group.

We show next that DOX is computationally correct. The proof relies on the fact that correctness remains intact unless the adversary finds one of the hidden PRP values, and the probability of the latter can be bounded by the *one-way* security of the obfuscator and the pseudorandomness of PRP.

Theorem 7 (DOX is computationally correct)**.** *Let* Obf *be an obfuscator and let* FE *be a computationally correct FE scheme. Then* $\mathsf{DOX}[\mathsf{FE}, \mathsf{PRP}, \mathsf{Obf}]$ *is computationally correct if the underlying PRP is pseudorandom and* Obf *is* OW *secure. More precisely, for any adversary* $\mathcal{A}$ *in game* CC *against* $\mathsf{DOX}[\mathsf{FE}, \mathsf{PRP}, \mathsf{Obf}]$*, placing at most* t *queries to* TGen*, there exist adversaries* $\mathcal{B}_1$*,* $\mathcal{B}_2$ *and* $\mathcal{B}_3$ *such that*

$$\mathbf{Adv}^{\mathsf{cc}}_{\mathsf{KS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{cc}}_{\mathsf{FE},\mathcal{B}_3}(\lambda) + (t+1) \cdot \mathbf{Adv}^{\mathsf{ow}}_{\mathsf{Obf},\mathcal{B}_2}(\lambda) + \mathbf{Adv}^{\mathsf{prp}}_{\mathsf{PRP},\mathcal{B}_1}(\lambda) + \frac{t+1}{|\mathsf{WSp}_\lambda|} .$$

Proof. The proof is simple and follows two game hops as follows.

Game$_0$: This is the CC game with respect to FE and PRP.

Game$_1$: In this game instead of a PRP a truly random permutation (simulated via lazy sampling) is used in the calculation of tokens in TGen oracle and preparing tp.

Game$_2$: In this game, if a token generation query or one of $\mathcal{A}$'s output words matches any of the randomly generated words (via lazy sampling) the game aborts.

The analyses of the game transitions are straightforward. The transition from Game$_0$ to Game$_1$ relies on the security of the PRP. The transition from Game$_1$ to Game$_2$ is down to the one-way security of the obfuscator (note that the only information leaked to the adversary about each of the random keywords is via an obfuscated circuit included in the extracted tokens). Finally, the advantage of the adversary in Game$_2$ can be bounded down to the correctness of FE. We give the details next.

Game$_0$ to Game$_1$. Any adversary $\mathcal{A}$ with visible advantage difference in these two games can be converted to an adversary $\mathcal{B}_1$ against the security of the PRP. Assume that lazy sampling is implemented using a table T, i.e., $T[\mathsf{w}]$ indicates the random value assigned to w. Algorithm $\mathcal{B}_1$ starts by generating an FE key pair. It handles a queries w of $\mathcal{A}$ to TGen by first computing $\mathsf{w}' \leftarrow \textsc{Fn}(\mathsf{w})$ via its Fn oracle, obfuscating the circuits associated with these keywords, and finally generating a token for the disjunction of the obfuscated circuits using the master secret key. Token generation after $\mathcal{A}$ terminates is handled similarly, and the remaining operations in of the CC game can be simulated using mpk. $\mathcal{B}_1$ will finally check if $\mathcal{A}$ succeeded in breaking correctness. If so, then its output will be 0. Else, it will be 1.

Note that when the Fn oracle implements the PRP, Game$_0$ is simulated for $\mathcal{A}$, and when it implements a random permutation Game$_1$ is simulated. A simple probability analysis yields,

$$\Pr[\mathrm{Game}_0(1^\lambda)] - \Pr[\mathrm{Game}_1(1^\lambda)] = \mathbf{Adv}^{\mathsf{prp}}_{\mathsf{PRP},\mathcal{B}_1}(\lambda) \ .$$

Game$_1$ to Game$_2$. Let us consider that the game is aborted if, at the end of the execution of Game$_2$, one considers all keywords explicitly output by the adversary (i.e., all w^* in the list of keywords queried from TGen plus the challenge keyword and message output by the adversary when it terminates), and for some keyword w in table T we have:

$$\mathsf{w}^* = T[\mathsf{w}] \ .$$

We bound the probability that this bad flag is set via the one-way security of the obfuscation. We build the required $\mathcal{B}_2$ against the $(t+1)$-OW security of Obf as follows. $\mathcal{B}_2$ first guesses the query i in which $\mathcal{A}$ first produces w by choosing an index $i \leftarrow_\$ [t+1]$, where t is an upper bound on the number of TGen queries that $\mathcal{A}$ makes and the extra 1 accounts for the challenge keyword it produces on termination. $\mathcal{B}_2$ then generates an FE key pair, runs $\mathcal{A}$ and answers its TGen queries using the master secret key and constructing $T[\mathsf{w}]$ as before, except when the i-th query comes (and all future w queries). In the latter case, $\mathcal{B}_2$ uses a new challenge obfuscated circuit it receives in the one-wayness game. Note that we have implicitly programmed $T[\mathsf{w}]$ to be an unknown value, which leads to an inconsistency with probability at most $(t+1)/|\mathsf{WSp}|$: an upper bound on the probability that this value collides with one of the values in T during the entire game. When the bad event is detected, and if $\mathcal{B}_2$'s guess was correct, then $\mathcal{B}_2$ can recognize the faulty keyword by checking the obfuscated circuits it received for a match, and it can win the one-wayness game. Hence,

$$\Pr[\mathrm{Game}_1(1^\lambda)] - \Pr[\mathrm{Game}_2(1^\lambda)] \leq (t+1) \cdot \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{Obf},t,\mathcal{B}_2}(\lambda) + \frac{t+1}{|\mathsf{WSp}_\lambda|} \ .$$

Analysis of Game$_2$. In this game we use $\mathcal{A}$ to build an adversary against the computational correctness of the underlying FE scheme. Note that if Game$_2$ does not abort, then $\mathsf{m} \neq T(\mathsf{w})$ when $\mathcal{A}$ terminates. We show that if $\mathcal{A}$ wins without any aborts we can build an adversary $\mathcal{B}_3$ which wins the FE correctness game. Algorithm $\mathcal{B}_3$ gets mpk and runs $\mathcal{A}$ on it. It answers $\mathcal{A}$'s TGen queries using its own oracle, still lazily sampling $T[\mathsf{w}]$ and asking for a trapdoor on the disjunction of the obfuscated circuits. When $\mathcal{A}$ returns (m, w), algorithm $\mathcal{B}_3$ also returns these. Note that this is winning pair iff it is a winning pair in the FE game. We therefore have

$$\Pr[\mathrm{Game}_2(1^\lambda)] = \mathbf{Adv}^{\mathrm{cc}}_{\mathsf{FE},\mathcal{B}_3}(\lambda) \ .$$

□

The proof of Res-PRIV security of this construction involves an intricate game hopping argument, in order to deal with all possible correlations allowed by the Res-PRIV model (which are the same as those allowed by full PRIV). We outline it below, highlighting how various ingredients are used in the construction, and provide a detailed proof in Appendix 8.

Theorem 8 (DOX is Res-PRIV)**.** *If* FE *is an* IND-CPA*-secure functional encryption scheme,* PRP *is a* PRP*-secure pseudorandom permutation family and* Obf *is a* DI*-secure obfuscator then scheme* DOX[FE, PRP, Obf] *is a* Res-PRIV*-secure keyword search scheme.*

Outline. The proof proceeds along six games as follows. Roughly speaking, after moving to a random permutation in Game$_1$ (and some bookkeeping in Game$_2$), in Game$_3$ we move from correlations between messages and keywords to their repetition patterns. In Game$_4$, we use obfuscation to deal with repetitions among keywords that do not match any of the messages (and were not queried to TGen in first phase). In Game$_5$, we use FE security to remove repetitions among messages that do not match any of the challenge keywords and were not queried to token-generation either (either due to legitimacy or adversarial restriction). Repetitions in all other cases can be dealt with using explicit values, the image matrix, or obfuscations. These steps make challenge ciphertexts independent of the challenge bit. In Game_6, using the security of the obfuscator we move to a setting where challenge tokens are also independent of the bit. In Game_6 advantage of any adversary is 0.

Game_0 : This game is identical to the Res-PRIV game.

Game_1 : Instead of PRP, a truly random permutation is used in TGen. We simulate the random permutation via a lazily sampled table T. This transition is sound down to PRP security.

Game_2 : We introduce a bad flag. We generate PRP values for all keywords and messages. If there are two T-values $(x_1, T(x_2))$ and $(x_2, T(x_2))$ such that $x_1 = T(x_2)$ we set bad. By the OW security of the obfuscator, these PRP values remain hidden and bad can only be set with negligible probability.

Game_3 : We compute the ciphertexts by encrypting $T(\mathbf{m}_b^*)$ instead of $\mathbf{m}_b^*$. This hop is reduced to the IND-CPA security of the FE, via explicit knowledge of challenge keywords and message by running the ppt sampler. Legitimacy will be violated if there is a w queried to TGen such that $\mathsf{w} = T(\mathsf{m}_b)$ or $\mathsf{m}_b = T(\mathsf{w})$. Both of these events set bad.

Game_4 : Call a challenge keyword *unpaired* if it was not any of the challenge messages, and *new* if it is not queried to first-phase TGen. In this game, instead of T values we use *forgetful* random values for all new and unpaired keywords. We bound this hop using DI. We simulate first-phase TGen using a lazily sampled T and a msk. Next, we run the Res-PRIV sampler *explicitly* and identify all new unpaired keywords. We define a DI sampler to sample consistent values on left and forgetful values on right (both independently of T), together with a second set consisting of sufficiently many consistent values on *both* sides. (The DI sampler does not need to respect any equality patterns.) This sampler can be shown to be statistically unpredictable. Once we receive the obfuscations, we use the first set, the explicit knowledge of challenge values and table T to form the challenge tokens and ciphertexts. For second-phase TGen queries we need to use consistent T values throughout. For values which match a first-phase query or a challenge messages we use T. If a query happens to *match a new unpaired keywords*—we can check this using the explicit knowledge of the keywords—we use a value from the second set of obfuscations. Otherwise we sample T values. We return 1 iff the adversary succeeds.

Game$_5$: Call a challenge message *unpaired* if it is not any of the challenge keywords, *LR-identical* if $\mathsf{m}_0^* = \mathsf{m}_1$, LR-differing if not equal, and *new* if it is unpaired and not queried to first-phase TGen. In this game instead of T values we use forgetful values for all unpaired LR-differing messages and all *new* LR-identical messages. We bound this hop down to IND-CPA. We will use the provided TGen oracle and only need to set T-values correctly. For first-phase TGen queries we lazily sample T. Next we run the sampler explicitly to obtain the challenges. For paired keywords or messages we use T-consistent values. For new unpaired keywords we use forgetful values (rule in Game$_4$). For unpaired messages, if LR-identical and queried to first-phase TGen (hence not new) we also use consistent T values. For LR-differing or new LR-identical messages we call LR in FE game, asking for T-consistent values on the left and independent forgetful values on the right. Note that LR-differing messages and new LR-identical messages are not queried to TGen at all due to our *restriction* on the adversary. If a second-phase TGen query matches a forgetful value generated in computing the LR query, we stop and guess that forgetful values were encrypted. (These values are information theoretically hidden if not encrypted.) Otherwise, we return 1 iff the adversary succeeds.

Game$_6$: In this game, irrespective of the bit, we use the second set of keywords for challenge token generation. We reduce this transition to the DI game. First-phase TGen queries are answered using a lazily sampled T and a generated msk. We set the DI sampler to run the PRIV sampler and on top of the output keywords, also ask for obfuscations of *messages* that are at the same time *LR-identical*, unpaired and new (it also outputs the random coins of first stage adversary, key generation and token extraction, along with a full image matrix as extra auxiliary information that will be needed for the second stage simulation). Using the symmetry of roles for keywords and messages in point functions, this sampler can be shown to be unpredictable whenever the PRIV sampler is. The obfuscations of messages will allow us to *check* if any of these messages (hidden under the obfuscation) match a first-phase TGEN query. We need this as according to the rules of Game_5 we must use T-consistent values. For paired messages, which we can find using the image matrix, we also use T-consistent values. For unpaired keywords we use forgetful values. For all other unpaired messages (be it LR-differing or never queried to TGEN) we use forgetful values (Game_5). Second phase TGEN queries are answered using T-consistent values relying on the fact that we can use the obfuscations to check matches with paired keywords and the restriction that adversary cannot query a new unpaired LR-identical messages to TGEN. We return 1 iff the adversary succeeds.

Challenge tokens in Game_6 are independent of the challenge bit. Due to the modifications in Game_4 and Game_5, the challenge ciphertexts are also independent of it. To see this note that ciphertexts contain on left and right: (1) identical T-consistent values that follow the correct repetition pattern for paired massages; (2) forgetful (independent) values for LR-differing messages; (3) identical T-consistent values that follow the correct repetition pattern for LR-identical messages queried in the first stage; (4) forgetful (independent) values for LR-identical messages not queried in the first stage. The adversary, therefore, has zero advantage in this game. □

6.5 The verifiably-obfuscate-encrypt-extract (VOEX) transform

We now present a fifth construction for point functions, which although simpler, conceptually relies on the observation that messages can be encoded as circuits that other circuits can evaluate. The obfuscator that we will rely on in our construction needs to be *verifiable*, meaning that there is an efficient algorithm to determine if a circuit $\overline{\mathsf{C}}$ is an obfuscation of a point function $\mathsf{C}[\mathsf{m}]$ for a message $\mathsf{m} \in \mathsf{MSp}_\lambda$. This property can be easily added by attaching a NIZK proof that there exist (m, r) such that $\overline{\mathsf{C}} = \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{m}]; \mathsf{r})$.

The VOEX transform. Let $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ be a non-interactive zero-knowledge proof system (see Section 2.3.4). Let Obf be an obfuscator supporting a circuit family $\mathsf{CSp} := \{\mathsf{CSp}^1_\lambda \cup \mathsf{CSp}^2_\lambda\}_{\lambda \in \mathbb{N}}$, where $\mathsf{CSp}^1_\lambda := \{\mathsf{C}[\mathsf{m}] : \mathsf{m} \in \mathsf{MSp}_\lambda\}$ and $\mathsf{CSp}^2_\lambda := \{\mathsf{D}[\mathsf{crs}, \mathsf{m}] : \mathsf{m} \in \mathsf{MSp}_\lambda, \mathsf{crs} \in [\mathsf{NIZK.Setup}(1^\lambda)]\}$ with

$$\mathsf{D}[\mathsf{crs}, \mathsf{w}](\overline{\mathsf{C}}, \pi) := \begin{cases} 1 & \text{if } \mathsf{NIZK.Verify}(\mathsf{crs}, \overline{\mathsf{C}}, \pi) \wedge \overline{\mathsf{C}}(\mathsf{w}) = 1 \ ; \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathsf{RSp} := \{\mathsf{RSp}_\lambda\}_{\lambda \in \mathbb{N}}$ denote the randomness space of Obf. Let FE be a functional encryption scheme supporting general circuits. We construct a keyword search scheme $\mathsf{KS} := \mathsf{VOEX}[\mathsf{FE}, \mathsf{NIZK}, \mathsf{Obf}]$ via the *Verifiably-Obfuscate-Encrypt-Extract* (VOEX) transform for keyword space $\mathsf{WSp} := \mathsf{MSp}$ as follows.

Setup: Algorithm $\mathsf{KS.Gen}(1^\lambda)$ generates a functional encryption key pair $(\mathsf{msk}, \mathsf{mpk}) \leftarrow_{\$} \mathsf{FE.Gen}(1^\lambda)$ and a common reference string $\mathsf{crs} \leftarrow_{\$} \mathsf{NIZK.Setup}(1^\lambda)$. It returns the key pair $((\mathsf{msk}, \mathsf{crs}), (\mathsf{mpk}, \mathsf{crs}))$.

Encryption: Algorithm $\mathsf{KS.Enc}((\mathsf{mpk}, \mathsf{crs}), \mathsf{m})$ generates $\overline{\mathsf{C}} \leftarrow_{\$} \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{m}]; \mathsf{r})$ for $\mathsf{r} \leftarrow_{\$} \mathsf{RSp}_\lambda$. It sets $\pi \leftarrow_{\$} \mathsf{NIZK.Prove}(\mathsf{crs}, \overline{\mathsf{C}}, (\mathsf{m}, \mathsf{r}))$ and finally returns $\mathsf{FE.Enc}(\mathsf{mpk}, (\overline{\mathsf{C}}, \pi))$.

Token generation: Algorithm $\mathsf{KS.TGen}((\mathsf{msk}, \mathsf{crs}), \mathsf{w})$ generates a token for the circuit $\mathsf{D}[\mathsf{crs}, \mathsf{w}]$ using the token-extraction algorithm $\mathsf{FE.TGen}$ and returns the result.

Evaluation: Algorithm $\mathsf{KS.Test}(\mathsf{c}, \mathsf{tk})$ simply runs $\mathsf{FE.Eval}(\mathsf{c}, \mathsf{tk})$.

Correctness of the construction follows from the correctness of the obfuscator and that of the functional encryption scheme, as well as the completeness of the proof system. Before presenting the theorem, we clarify the requirements on the underlying obfuscation scheme.

PRIV-restricted samplers. As shown in the work of Barak et al. [BGI+01], no 2-circuits general-purpose VBB obfuscator exists. This impossibility result can be extended to rule out general-purpose DI obfuscation as well and, in particular, DI obfuscation supporting the class of circuits we require for instantiating our construction above. Briefly, consider the circuits $\mathsf{D}[\mathsf{w}](\mathsf{C}) := \mathsf{C}(\mathsf{w})$ and a sampler $\mathcal{S}$ that outputs circuits $(\mathsf{D}[\mathsf{w}], \mathsf{C}[\mathsf{w}])$ on the left and $(\mathsf{D}[\mathsf{w}], \mathsf{C}[\overline{\mathsf{w}}])$ on the right for a uniform keyword w. This sampler can be shown to be unpredictable. However, the DI game can be won by evaluating (an obfuscation of) the first challenge circuit on an obfuscation of the second challenge circuit.

For our particular construction, however, we rely on a weaker form of obfuscation that is only required to support samplers that output circuits and messages that are restricted by the PRIV legitimacy condition (this is a result of our reduction strategy). Concretely, such circuits and messages will result on image matrices that are identical on the left and right, which completely rules out attacks akin to those in [BGI+01]. We call this class of DI samplers PRIV-*restricted*. Formally, a DI sampler $\mathcal{S}$ is PRIV-restricted for circuit class CSp if for a legitimate PRIV sampler $\mathcal{S}'$ and a non-interactive zero-knowledge proof system NIZK it operates as follows.

$$\begin{aligned}\mathcal{S}(1^\lambda, st, \mathsf{crs}):\quad & (\mathbf{w}_0, \mathbf{w}_1, \mathbf{m}_0, \mathbf{m}_1, z) \leftarrow_{\$} \mathcal{S}'(1^\lambda, st);\\ & \text{if } \mathsf{crs} \notin [\mathsf{NIZK.Setup}(1^\lambda)] \text{ return } ([], [], \epsilon)\\ & \text{else return } ((\mathsf{D}[\mathsf{crs}, \mathbf{w}_0], \mathsf{C}[\mathbf{m}_0]), (\mathsf{D}[\mathsf{crs}, \mathbf{w}_1], \mathsf{C}[\mathbf{m}_1]), (z, \mathsf{C}[\mathbf{w}_0](\mathbf{m}_0)))\end{aligned}$$

The PRIV security of the VOEX construction is established in the following theorem.

Theorem 9 (VOEX is PRIV secure)**.** *If* FE *is* IND-CPA *secure,* NIZK *is perfectly sound and computationally zero-knowledge, and obfuscator* Obf *is* DI *secure with respect to* PRIV*-restricted samplers, then scheme* $\mathsf{VOEX}[\mathsf{FE}, \mathsf{Obf}, \mathsf{NIZK}]$ *is* PRIV *secure.*

Proof (Outline). The proof follows a sequence of three game hops.

Game$_0$: This is the PRIV game for the VOEX construction.

Game$_1$: We say a message $\mathbf{m}_b[i]$ is *unpaired* if $\mathbf{m}_b[i] \notin \mathbf{w}_b$. Note that for all legitimate samplers if $\mathbf{m}_0[i]$ is unpaired, so is $\mathbf{m}_1[i]$. In this game, the LR oracle replaces all unpaired messages (on both sides) which are LR-differing (that is, when $\mathbf{m}_0[i] \neq \mathbf{m}_0[i]$) with random and independently sampled values. The distance to the previous game can be upper bounded using the IND-CPA security of the FE scheme. The legitimacy of the algorithm playing the IND-CPA game in the reduction is guaranteed because: (1) replaced messages are LR-differing and therefore the adversary cannot ask tokens for those in the PRIV game (and hence also the IND-CPA hame); (2) replacements are random and information-theoretically hidden from the adversary when the original messages are encrypted, and if the adversary asks for token for one of the random replacements, it can be only because the ciphertexts are leaking one of these replacements.

Game$_2$: In this game we use Sim to generate simulated proofs in the LR oracle without using the explicit knowledge of the messages. The distance to the previous game can be bounded by the zero-knowledge property of the NIZK proof system.

Game$_3$: In this game, regardless of bit b, we use the second set of keywords and messages to generate the challenge. We reduce this transition to the DI game. We set the DI sampler to take a crs along with the state st required to run the PRIV sampler; it runs the PRIV sampler to obtain keywords and messages, and it outputs a D circuit (with a hardwired crs) for every keyword and a C circuit for every message (after carefully replacing LR-differing unpaired messages with random values as in Game$_1$). This DI sampler is by definition PRIV-restricted and it is unpredictable whenever the underlying PRIV sampler is unpredictable. The proof of this fact relies on the perfect soundness of the proof system under the binding crs, whose validity we assume can be efficiently checked [GS08]. The PRIV predictor uses its oracle to answer the DI predictor's queries (if a query contains an obfuscated point circuit and the attached proof verifies, the unbounded PRIV predictor can reverse-engineer the obfuscated circuit to recover its underlying point, and query its own oracle on it). The adversary against the DI game simulates the environment of Game$_2$ as follows. It generates a key pair $(\mathsf{msk}, \mathsf{mpk})$ and simulated $(\mathsf{crs}, \mathsf{tp})$ for the NIZK, and then it runs the first stage of the PRIV adversary until it obtains state st for the LR oracle call. It then calls its own LR oracle on (st, crs), obtaining a set of obfuscations. As before, the trapdoor tp is used to produce simulated proofs of the obfuscations of C circuits corresponding to messages, resulting in well-formed challenge ciphertexts. It then runs the second stage of the PRIV adversary, answering its token extraction queries using the master secret key and, when this adversary returns a bit b', it uses it as its own guess.

In Game$_3$, the challenge is independent of bit b and therefore the adversary has zero advantage. $\square$

Chapter 7

Updatable Functional Encryption

The results described in this section have been published in [AIT16].

The concept of functional encryption (FE), a generalization of identity-based encryption, attribute-based encryption, inner-product encryption and other forms of public-key encryption, was independently formalized by Boneh, Sahai and Waters [BSW11] and O'Neil [O'N10]. In 2013, Garg et al. [GGH$^+$13] put forth the first candidate construction of an FE scheme supporting all polynomial-size circuits based on indistinguishability obfuscation (iO), which is now known as a central hub for the realization of many cryptographic primitives [SW14].

The most common approach is to model functions as circuits. In some works, however, functions are modeled as Turing machines (TM) or random-access machines (RAM). Recently, Ananth and Sahai [AS16] constructed an adaptively secure functional encryption scheme for TM, based on indistinguishability obfuscation. Nonetheless, their work does not tackle the problem of having the token update the encrypted message, over which other tokens can be subsequently executed.

In the symmetric setting, the notion of garbled RAM, introduced by Lu and Ostrovsky [LO13] and revisited by Gentry et al. [GHL$^+$14], addresses this important use-case where garbled memory data can be reused across multiple program executions. Garbled RAM can be seen as an analogue of Yao's garbled circuits [Yao86] (see also [BHR12] for an abstract generalization) that allows a user to garble a RAM program without having to compile it into a circuit first. As a result, the time it takes to evaluate a garbled program is only proportional to the running time of the program on a random-access machine. Several other candidate constructions were also proposed in [GHRW14, CHJV14, CHJV15, CH16].

Desmedt et al. [DIPV14] proposed an FE with controlled homomorphic properties. However, their scheme updates and re-encrypts the entire data, which carries a highly inefficient evaluation-time.

Our contribution. In this chapter, we propose a new primitive that we call *updatable functional encryption* (UFE). It bears resemblance to functional encryption in that encryption is carried out in the public-key setting and the owner of the master secret key can issue tokens for functions—here, modeled as RAM programs—of its choice that allow learning the outcome of the function on the message underneath a ciphertext. We envision tokens that are also capable to *update* the ciphertext, over which other tokens can be subsequently executed. We impose strict efficiency constrains in that the run-time of a token $\overline{\mathsf{P}}$ on ciphertext CT is proportional to the run-time of its clear-form counterpart (program P on memory D) up to a *polylogarithmic* factor in the size of D. We define a security notion for our primitive and propose a candidate construction based on an instance of distributional indistinguishability (DI) obfuscation. Recent results put differing-inputs obfuscation (diO) [ABG$^+$13] with auxiliary information in contention with other assumptions [BST16]; one might question if similar attacks apply to the obfuscation notion we require in our reduction. As far as we can tell, the answer is negative. However, we view our construction as a starting point towards the realization of other updatable functional encryption schemes from milder forms of obfuscation.

7.1 RAM programs

In the RAM model of computation, a program P has random-access to some initial *memory data* D, comprised of $|\mathsf{D}|$ *memory cells*. At each *CPU step* of its execution, P reads from and writes to a single memory cell *address*, which is determined by the previous step, and updates its internal state. By convention, the address in the first step is set to the first memory cell of D, and the initial internal state is empty. Only when P reaches the final step of its execution, it outputs a result y and terminates. We use the notation $\mathsf{y} \leftarrow \mathsf{P}^{\mathsf{D}\rightarrow\mathsf{D}^\star}$ to indicate this process, where $\mathsf{D}^\star$ is the resulting memory data when P terminates, or simply $\mathsf{y} \leftarrow \mathsf{P}^{\mathsf{D}}$ if we don't care about the resulting memory data. We also consider the case where the memory data *persists* between a sequential execution of n programs, and use the notation $(\mathsf{y}_1, ..., \mathsf{y}_n) \leftarrow (\mathsf{P}_1, ..., \mathsf{P}_n)^{\mathsf{D}\rightarrow\mathsf{D}^\star}$ as short for $(\mathsf{y}_1 \leftarrow \mathsf{P}_1^{\mathsf{D}\rightarrow\mathsf{D}_1}\ ;\ ...\ ;\ \mathsf{y}_n \leftarrow \mathsf{P}_n^{\mathsf{D}_{n-1}\rightarrow\mathsf{D}^\star})$. In more detail, a RAM program description is a 4-tuple $\mathsf{P} := (\mathcal{Q}, \mathcal{T}, \mathsf{OSp}, \delta)$, where:

- $\mathcal{Q}$ is the set of all possible states, which always includes the empty state ϵ.
- $\mathcal{T}$ is the set of all possible contents of a memory cell. If each cell contains a single bit, $\mathcal{T} = \{0, 1\}$.
- OSp is the output space of P, which always includes the empty output ϵ.
- δ is the transition function, modeled as a circuit, which maps $(\mathcal{Q} \times \mathcal{T})$ to $(\mathcal{T} \times \mathcal{Q} \times \mathbb{N} \times \mathsf{OSp})$. On input an internal state $st_i \in \mathcal{Q}$ and a content of a memory cell $\mathsf{read}_i \in \mathcal{T}$, it outputs a (possibly different) content of a memory cell $\mathsf{write}_i \in \mathcal{T}$, an internal state $st_{i+1} \in \mathcal{Q}$, an address of a memory cell $\mathsf{addr}_{i+1} \in \mathbb{N}$ and an output $\mathsf{y} \in \mathsf{OSp}$.

In Figure 7.1 we show how program P is executed on a random-access machine with initial memory data D.

To conveniently specify the *efficiency* and *security* properties of the primitive we propose in the following section, we define functions $\mathsf{runTime}$ and $\mathsf{accessPattern}$ that on input a program P and some initial memory data D return the number of steps required for P to complete its execution on D and the list of addresses accessed during the execution, respectively. In other words, as per description in Figure 7.1, $\mathsf{runTime}$ returns the value i when P terminates, whereas $\mathsf{accessPattern}$ returns List. More generally, we also allow these functions to receive as input a *set* of programs $(\mathsf{P}_1, ..., \mathsf{P}_n)$ to be executed sequentially on persistent memory, initially set to D.

```
EXECUTE $\mathsf{P}^{\mathsf{D}}$:
$i \leftarrow 0$;   $\mathsf{addr}_i \leftarrow 0$;   $st_i \leftarrow \epsilon$;   $\mathsf{y} \leftarrow \epsilon$;   $\mathsf{List} \leftarrow []$
while ($\mathsf{y} = \epsilon$)
    // step $i$
    $\mathsf{List} \leftarrow \mathsf{addr}_i : \mathsf{List}$ // record the access pattern
    $\mathsf{read}_i \leftarrow \mathsf{D}[\mathsf{addr}_i]$ // read from memory
    $(\mathsf{write}_i, st_{i+1}, \mathsf{addr}_{i+1}, \mathsf{y}) \leftarrow \delta(st_i, \mathsf{read}_i)$
    $\mathsf{D}[\mathsf{addr}_i] \leftarrow \mathsf{write}_i$ // write to memory
    $i \leftarrow i + 1$
return ($\mathsf{y}$)
```

Figure 7.1: Execution process of a program P on a RAM machine with memory D.

7.2 Definitions

We propose a new primitive that we call *updatable functional encryption.* It bears resemblance to functional encryption in that encryption is carried out in the public-key setting and the owner of the master secret key can issue tokens for functions of its choice that allows the holder of the token to learn the outcome of the function on the message underneath a ciphertext. Here, we model functions as RAM programs instead of circuits, which is closer to how programs are expressed in von Neumann architecture and avoids the RAM-to-circuit compilation. Not only that, we envision tokens that are capable to *update* the ciphertext, over which other tokens can be subsequently executed. Because the ciphertext evolves every time a token is executed and for better control over what information is revealed, each token is numbered sequentially so that it can only be executed *once* and *after all previous extracted tokens* have been executed on that ciphertext. Informally, the security requires that the ciphertext should not reveal more than what can be learned by applying the extracted tokens in order. As for efficiency, we want the run-time of a token to be proportional to the run-time of the program up to a *polylogarithmic* factor in the length of the encrypted message.

SYNTAX. An updatable functional encryption scheme UFE for program family $\mathcal{P} := \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$ with message space $\mathsf{MSp} := \{\mathsf{MSp}_\lambda\}_{\lambda\in\mathbb{N}}$ is specified by three ppt algorithms as follows.

- $\mathsf{UFE.Setup}(1^\lambda)$ is the setup algorithm and on input a security parameter 1^λ it outputs a master secret key msk and a master public key mpk;
- $\mathsf{UFE.TokenGen}(\mathsf{msk}, \mathsf{P}, \mathsf{tid})$ is the token-generation algorithm and on input a master secret key msk, a program description $\mathsf{P} \in \mathcal{P}_\lambda$ and a token-id $\mathsf{tid} \in \mathbb{N}$, outputs a token (i.e. another program description) $\overline{\mathsf{P}}_{\mathsf{tid}}$;
- $\mathsf{UFE.Enc}(\mathsf{mpk}, \mathsf{D})$ is the encryption algorithm and on input a master public key mpk and memory data $\mathsf{D} \in \mathsf{MSp}_\lambda$ outputs a ciphertext CT.

We do not explicitly consider an evaluation algorithm. Instead, the RAM program $\overline{\mathsf{P}}$ output by $\mathsf{UFE.TokenGen}$ executes directly on memory data CT, a ciphertext resulting from the $\mathsf{UFE.Enc}$ algorithm. Note that this brings us close to the syntax of Garbled RAM, but in contrast encryption is carried out in the public-key setting.

Correctness. We say that UFE is correct if for every security parameter $\lambda \in \mathbb{N}$, for every memory data $\mathsf{D} \in \mathsf{MSp}_\lambda$ and for every sequence of polynomial length in λ of programs $(\mathsf{P}_1, ..., \mathsf{P}_n)$, it holds that

$$\Pr\left[\mathsf{y}_1 = \mathsf{y}'_1 \wedge ... \wedge \mathsf{y}_n = \mathsf{y}'_n \;\middle|\; \begin{array}{l} (\mathsf{msk}, \mathsf{mpk}) \leftarrow_{\$} \mathsf{UFE.Setup}(1^\lambda) \\ \mathsf{CT} \leftarrow_{\$} \mathsf{UFE.Enc}(\mathsf{mpk}, \mathsf{D}) \\ \text{for } i \in [n] \\ \quad \overline{\mathsf{P}}_i \leftarrow_{\$} \mathsf{UFE.TokenGen}(\mathsf{msk}, \mathsf{P}_i, i) \\ (\mathsf{y}_1, ..., \mathsf{y}_n) \leftarrow (\mathsf{P}_1, ..., \mathsf{P}_n)^{\mathsf{D}} \\ (\mathsf{y}'_1, ..., \mathsf{y}'_n) \leftarrow (\overline{\mathsf{P}}_1, ..., \overline{\mathsf{P}}_n)^{\mathsf{CT}} \end{array}\right] = 1.$$

Efficiency. Besides the obvious requirement that all algorithms run in polynomial-time in the length of their inputs, we also require that the run-time of token $\overline{\mathsf{P}}$ on ciphertext CT is proportional to the run-time of its clear-form counterpart (program P on memory D) up to a polynomial factor in λ and up to a polylogarithmic factor in the length of D. More precisely, we require that for every $\lambda \in \mathbb{N}$, for every sequence of polynomial length in λ of programs $(\mathsf{P}_1, ..., \mathsf{P}_n)$ and every memory data $\mathsf{D} \in \mathsf{MSp}_\lambda$, there exists a fixed polynomial function poly and a fixed polylogarithmic function $\mathsf{polylog}$ such that

$$\Pr\left[\begin{array}{c} \mathsf{runTime}((\overline{\mathsf{P}}_1, ..., \overline{\mathsf{P}}_n), \mathsf{CT}) \leq \\ \mathsf{runTime}((\mathsf{P}_1, ..., \mathsf{P}_n), \mathsf{D}) \cdot \\ \mathsf{poly}(\lambda) \cdot \mathsf{polylog}(|\mathsf{D}|) \end{array} \middle| \begin{array}{l} (\mathsf{msk}, \mathsf{mpk}) \leftarrow_{\$} \mathsf{UFE.Setup}(1^\lambda) \\ \mathsf{CT} \leftarrow_{\$} \mathsf{UFE.Enc}(\mathsf{mpk}, \mathsf{D}) \\ \text{for } i \in [n] \\ \quad \overline{\mathsf{P}}_i \leftarrow_{\$} \mathsf{UFE.TokenGen}(\mathsf{msk}, \mathsf{P}_i) \end{array}\right] = 1.$$

In particular, this means that for a program P running in sublinear-time in $|\mathsf{D}|$, the run-time of $\overline{\mathsf{P}}$ over the encrypted data remains sublinear.

Security. Let UFE be an updatable functional encryption scheme. We say UFE is *selectively secure* if for any legitimate ppt adversary $\mathcal{A}$

$$\mathbf{Adv}^{\text{sel}}_{\mathsf{UFE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\text{SEL}^{\mathcal{A}}_{\mathsf{UFE}}(1^\lambda)\right] - 1 \in \text{Negl},$$

where game $\text{SEL}^{\mathcal{A}}_{\mathsf{UFE}}(1^\lambda)$ is defined in Figure 7.2. We say $\mathcal{A}$ is legitimate if the following two conditions are satisfied:

1. $(\mathsf{P}_1, ..., \mathsf{P}_n)^{\mathsf{D}_0} = (\mathsf{P}_1, ..., \mathsf{P}_n)^{\mathsf{D}_1}$
2. $\mathsf{accessPattern}((\mathsf{P}_1, ..., \mathsf{P}_n), \mathsf{D}_0) = \mathsf{accessPattern}((\mathsf{P}_1, ..., \mathsf{P}_n), \mathsf{D}_1)$

These conditions avoid that the adversary trivially wins the game by requesting tokens whose output differ on left and right challenge messages or have different access patterns.

```
$\underline{\text{SEL}^{\mathcal{A}}_{\mathsf{UFE}}(1^\lambda)}$:
$(\mathsf{D}_0, \mathsf{D}_1, (\mathsf{P}_1, ..., \mathsf{P}_n), st) \leftarrow_{\$} \mathcal{A}_0(1^\lambda)$
$(\mathsf{msk}, \mathsf{mpk}) \leftarrow_{\$} \mathsf{UFE.Setup}(1^\lambda)$
$b \leftarrow_{\$} \{0,1\}$
$\mathsf{CT} \leftarrow_{\$} \mathsf{UFE.Enc}(\mathsf{mpk}, \mathsf{D}_b)$
for $i \in [n]$
  $\overline{\mathsf{P}}_i \leftarrow_{\$} \mathsf{UFE.TokenGen}(\mathsf{msk}, \mathsf{P}_i)$
$b' \leftarrow_{\$} \mathcal{A}_1(\mathsf{CT}, (\overline{\mathsf{P}}_1, ..., \overline{\mathsf{P}}_n), st)$
return $(b = b')$
```

Figure 7.2: Selective security of an updatable functional encryption scheme UFE.

7.3 Candidate construction

The idea of our construction is the following. Before encryption we append to the cleartext the token-id of the first token to be issued, the address of the first position to be read and the initial state of the program. These values are all pre-defined at the beginning. We then split the data into bits and label each of them with a common random tag, their position on the array and a counter that keeps track of how many times that bit was updated (initially 0). Then, we build a Merkle tree over the labeled bits. Later, this will allow us to check the consistency of the data without having to read through all of it. It also binds a token-id, a read-position and a state to the data at a particular stage. Finally, we encrypt each node of the tree, twice, and attach a NIZK proof attesting that they encrypt the same content. Tokens include the decryption key inside their transition circuit in order to perform the computation over the clear data and re-encrypt the nodes at the end of each CPU step. These circuits are obfuscated to protect the decryption key, and the random coins necessary to re-encrypt come from a puncturable PRF. The proof then follows a mix of different strategies seen in [NY90, IPS15, GGH+13, ABF16, GJKS15].

- $\mathsf{UFE.Setup}(1^\lambda)$ samples public-key encryption key pairs $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow_\$ \mathsf{PKE.Setup}(1^\lambda)$ and $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow_\$ \mathsf{PKE.Setup}(1^\lambda)$, a common reference string $\mathsf{crs} \leftarrow_\$ \mathsf{NIZK.Setup}(1^\lambda)$ and a collision-resistant hash function $\mathsf{H} \leftarrow_\$ \mathsf{H}_\lambda$. It then sets constants (l_1, l_2, l_3) as the maximum length of token-ids, addresses and possible states induced by the supported program set $\mathcal{P}_\lambda$, respectively, encoded as bit-strings. Finally, it sets $\mathsf{msk} \leftarrow \mathsf{sk}_0$ and $\mathsf{mpk} \leftarrow (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{crs}, \mathsf{H}, (l_1, l_2, l_3))$ and outputs the key pair $(\mathsf{msk}, \mathsf{mpk})$.

- $\mathsf{UFE.Enc}(\mathsf{mpk}, \mathsf{D})$ parses mpk as $(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{crs}, \mathsf{H}, (l_1, l_2, l_3))$ and appends to the memory data D the token-id 1, address 0 and the empty state ϵ, encoded as bit-stings of length l_1, l_2 and l_3, respectively: $\mathsf{D} \leftarrow (\mathsf{D}, 1, 0, \epsilon)$. (We assume from now on that $|\mathsf{D}|$ is a power of 2. This is without loss of generality since D can be padded.) $\mathsf{UFE.Enc}$ sets $\mathsf{z} \leftarrow \log(|\mathsf{D}|)$, samples a random string $\mathsf{tag} \leftarrow_\$ \{0,1\}^\lambda$ and constructs a perfectly balanced binary tree $\mathsf{T} := \{\mathsf{node}^{(i,j)}\}$, where leafs are set as

$$\forall j \in \{0, ..., (|\mathsf{D}| - 1)\},\ \mathsf{node}^{(\mathsf{z},j)} \leftarrow (\mathsf{D}[j], \mathsf{tag}, (\mathsf{z}, j), 0)$$

and intermediate nodes are computed as

$$\begin{aligned}&\forall i \in \{(\mathsf{z}-1), ..., 0\},\ \forall j \in \{0, ..., (2^i-1)\},\\ &\quad \mathsf{node}^{(i,j)} \leftarrow (\mathsf{H}(\mathsf{node}^{(i+1,2j)}, \mathsf{node}^{(i+1,2j+1)})).\end{aligned}$$

$\mathsf{UFE.Enc}$ then encrypts each node independently under pk_0 and pk_1, i.e.

$$\begin{aligned}&\forall i \in \{0, ..., \mathsf{z}\},\ \forall j \in \{0, ..., (2^i-1)\},\\ &\quad \mathsf{r}_0^{(i,j)} \leftarrow_{\$} \mathsf{RSp}_\lambda\ ;\ \mathsf{r}_1^{(i,j)} \leftarrow_{\$} \mathsf{RSp}_\lambda\\ &\quad \mathsf{CT}_0^{(i,j)} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_0, \mathsf{node}^{(i,j)}; \mathsf{r}_0^{(i,j)})\\ &\quad \mathsf{CT}_1^{(i,j)} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_1, \mathsf{node}^{(i,j)}; \mathsf{r}_1^{(i,j)})\end{aligned}$$

and computes NIZK proofs that $\mathsf{CT}_0^{(i,j)}$ and $\mathsf{CT}_1^{(i,j)}$ encrypt the same content. More precisely,

$$\begin{aligned}&\forall i \in \{0, ..., \mathsf{z}\},\ \forall j \in \{0, ..., (2^i-1)\},\\ &\quad \pi^{(i,j)} \leftarrow_{\$} \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{x}^{(i,j)}, (\mathsf{node}^{(i,j)}, \mathsf{r}_0^{(i,j)}, \mathsf{r}_1^{(i,j)})),\end{aligned}$$

where $\mathsf{x}^{(i,j)}$ is the NP statement

$$\exists(\mathsf{m}, \mathsf{r}_0, \mathsf{r}_1) : \mathsf{CT}_0^{(i,j)} = \mathsf{PKE.Enc}(\mathsf{pk}_0, \mathsf{m}; \mathsf{r}_0) \wedge \mathsf{CT}_1^{(i,j)} = \mathsf{PKE.Enc}(\mathsf{pk}_1, \mathsf{m}; \mathsf{r}_1).$$

Finally, $\mathsf{UFE.Enc}$ lets

$$\mathsf{CT} := \{(\mathsf{CT}_0^{(i,j)}, \mathsf{CT}_1^{(i,j)}, \pi^{(i,j)})\},$$

which encodes a perfectly balanced tree, and outputs it as a ciphertext of memory data D under mpk.

- $\mathsf{UFE.TokenGen}(\mathsf{msk}, \mathsf{mpk}, \mathsf{P}, \mathsf{tid})$ parses $(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{crs}, \mathsf{H}, (l_1, l_2, l_3)) \leftarrow \mathsf{mpk}$, $\mathsf{sk}_0 \leftarrow \mathsf{msk}$ and $(\mathcal{Q}, \mathcal{T}, \mathsf{OSp}, \delta) \leftarrow \mathsf{P}$. Then, a new puncturable PRF key $\mathsf{k} \leftarrow_{\$} \mathsf{PPRF.Gen}(1^\lambda)$ is sampled. Next, it sets a circuit $\widehat{\delta}$ as described in Figure 7.3, using the parsed values as the appropriate hardcoded constants with the same naming. $\mathsf{UFE.TokenGen}$ then obfuscates this circuit by computing $\overline{\delta} \leftarrow_{\$} \mathsf{Obf}(\widehat{\delta})$. Finally, for simplicity in order to avoid having to explicitly deal with the data structure in the ciphertext, and following a similar approach as in [CCHR15], we define token $\overline{\mathsf{P}}$ not by its transition function, but by pseudocode, as the RAM program that executes on CT the following:

1. Set initial state $st \leftarrow \epsilon$, initial address $\mathsf{addr} \leftarrow 0$ and empty output $\mathsf{y} \leftarrow \epsilon$.
2. While $(\mathsf{y} = \epsilon)$
   (a) Construct a tree $\overline{\mathsf{T}}$ by selecting from CT the leaf at address addr and the last $(l_1 + l_2 + l_3)$ leafs (that should encode tid, addr and st if CT is valid), as well as all the necessary nodes to compute the hash values of their path up to the root.
   (b) Evaluate $(\overline{\mathsf{T}}, \mathsf{addr}, \mathsf{y}) \leftarrow \overline{\delta}(\overline{\mathsf{T}})$.
   (c) Update CT by writing the resulting $\overline{\mathsf{T}}$ to it.
3. Output y.

Theorem 10. *Let* PKE *be an* IND-CCA *secure public-key encryption scheme, let* NIZK *be a non-interactive zero knowledge proof system with perfect completeness, computational zero knowledge and statistical soundness, let* H *be a collision-resistant hash function family, let* PPRF *be a puncturable pseudorandom function and let* Obf *be an iO-secure obfuscator that is also DI-secure w.r.t. the class of samplers described in* Game$_4$*. Then, the updatable functional encryption scheme* $\mathsf{UFE}[\mathsf{PKE}, \mathsf{NIZK}, \mathsf{H}, \mathsf{PPRF}, \mathsf{Obf}]$ *detailed in Section 7.3 is selectively secure (as per definition in Figure 7.2).*

Proof (Outline). The proof proceeds via a sequence of games as follows.

Game$_0$: This game is identical to the real SEL game when the challenge bit $b = 0$, i.e. the challenger encrypts D_0 in the challenge ciphertext.

Game$_1$: In this game, the common reference string and NIZK proofs are simulated. More precisely, at the beginning of the game, the challenger executes $(\mathsf{crs}, \mathsf{tp}) \leftarrow_{\$} \mathsf{Sim}_0(1^\lambda)$ to produce the crs that is included in the mpk, and proofs in the challenge ciphertext are computed with Sim_1 and tp. The distance to the previous game can be bounded by the zero-knowledge property of NIZK.

Game$_2$: Let $\mathsf{T}_0 := \{\mathsf{node}_0^{(i,j)}\}$ be the perfectly balanced tree resulting from the encoding of D_0 with tag_0, and $\mathsf{T}_1 := \{\mathsf{node}_1^{(i,j)}\}$ the one resulting from the encoding of D_1 with tag_1, where $(\mathsf{D}_0, \mathsf{D}_1)$ are the challenge messages queried by the adversary and

Hardcoded: Transition circuit δ, token-id tid^*, secret key sk_0, puncturable PRF key k, public keys pk_0 and pk_1, common reference string crs, hash function H and bit-length constants (l_1, l_2, l_3).
Input: Tree $\overline{\mathsf{T}}$.

Execute these 7 steps:

1. Verify the NIZK proof in each node of tree $\overline{\mathsf{T}}$, and decrypt the first ciphertext of each node with sk_0. Let T be the resulting decrypted tree.

```
$\forall (i,j) \in \mathbb{N}^2 : \overline{\mathsf{node}}^{(i,j)} \in \overline{\mathsf{T}}$,
    parse $\overline{\mathsf{node}}^{(i,j)}$ as $(\mathsf{CT}_0^{(i,j)}, \mathsf{CT}_1^{(i,j)}, \pi^{(i,j)})$ or return $\perp$
    if $\mathsf{NIZK.Verify}(\mathsf{crs}, \mathsf{x}^{(i,j)}, \pi^{(i,j)}) = \mathsf{false}$ return $\perp$
    $\mathsf{node}^{(i,j)} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_0, \mathsf{CT}_0^{(i,j)})$
let $\mathsf{T} := \{\mathsf{node}^{(i,j)}\}$
```

2. On the decrypted tree T, verify the path of each leaf up to the root (i.e. intermediate nodes must be equal to the hash of their children) and check that all leafs are marked with the same random tag.

```
$\mathsf{z} \leftarrow \max\{i \in \mathbb{N} : \mathsf{node}^{(i,j)} \in \mathsf{T}, \exists j \in \mathbb{N}\}$
$\forall j \in \mathbb{N} : \mathsf{node}^{(\mathsf{z},j)} \in \mathsf{T}$,
    $\forall i \in \{(\mathsf{z}-1), ..., 0\}$
        if $\mathsf{node}^{(i, \lfloor \frac{j}{2^{(\mathsf{z}-i)}} \rfloor)} \neq \mathsf{H}(\mathsf{node}^{((i+1), 2\lfloor \frac{j}{2^{(\mathsf{z}-i)}} \rfloor)}, \mathsf{node}^{((i+1), (2\lfloor \frac{j}{2^{(\mathsf{z}-i)}} \rfloor + 1))})$ return $\perp$
    parse $\mathsf{node}^{(\mathsf{z},j)}$ as $(\mathsf{value}^{(\mathsf{z},j)}, \mathsf{tag}^{(\mathsf{z},j)}, \mathsf{position}^{(\mathsf{z},j)}, \mathsf{counter}^{(\mathsf{z},j)})$ or return $\perp$
if $\exists (j,j') \in \mathbb{N}^2 : \mathsf{node}^{(\mathsf{z},j)} \in \mathsf{T} \wedge \mathsf{node}^{(\mathsf{z},j')} \in \mathsf{T} \wedge \mathsf{tag}^{(\mathsf{z},j)} \neq \mathsf{tag}^{(\mathsf{z},j')}$ return $\perp$
```

3. Read the token-id, address and state of the current step encoded in tree T. Check that the token-id matches the one hardcoded in this token. Then, evaluate the transition circuit δ.

```
read $(\mathsf{tid}, \mathsf{addr}, st)$ with fixed bit-length $(l_1, l_2, l_3)$ from $\mathsf{T}$ or return $\perp$
if $\mathsf{tid} \neq \mathsf{tid}^*$ return $\perp$
$(\mathsf{value}^{(\mathsf{z},\mathsf{addr})}, st, \mathsf{addr}, \mathsf{y}) \leftarrow \delta(st, \mathsf{value}^{(\mathsf{z},\mathsf{addr})})$
```

4. If the transition circuit δ outputs some result y then increase the token-id and reset the internal state and address.

```
if $\mathsf{y} \neq \epsilon$ then $\mathsf{tid} \leftarrow \mathsf{tid} + 1$ ; $st \leftarrow 0$ ; $\mathsf{addr} \leftarrow 0$
```

5. Write the (possibly new) token-id, address and state to tree T, update the counters of leaf nodes and recompute the path of each leaf up to the root.

```
write $(\mathsf{tid}, \mathsf{addr}, st)$ with fixed bit-length $(l_1, l_2, l_3)$ to $\mathsf{T}$
$\forall j \in \mathbb{N} : \mathsf{node}^{(\mathsf{z},j)} \in \mathsf{T}$,
    $\mathsf{counter}^{(\mathsf{z},j)} \leftarrow \mathsf{counter}^{(\mathsf{z},j)} + 1$
$\forall j \in \mathbb{N} : \mathsf{node}^{(\mathsf{z},j)} \in \mathsf{T}$,
    $\forall i \in \{(\mathsf{z}-1), ..., 0\}$
        $\mathsf{node}^{(i, \lfloor \frac{j}{2^{(\mathsf{z}-i)}} \rfloor)} \leftarrow \mathsf{H}(\mathsf{node}^{((i+1), 2\lfloor \frac{j}{2^{(\mathsf{z}-i)}} \rfloor)}, \mathsf{node}^{((i+1), (2\lfloor \frac{j}{2^{(\mathsf{z}-i)}} \rfloor + 1))})$
```

6. Re-encrypt all nodes of T (as before, encrypt under pk_0 and pk_1 and add NIZK proofs under crs). To extract the necessary random coins, we use the puncturable PRF under key k, providing as input the input of this circuit, i.e. $\overline{\mathsf{T}}$.

```
$\forall (i,j) \in \mathbb{N}^2 : \mathsf{node}^{(i,j)} \in \mathsf{T}$,
    $(\mathsf{r}_0^{(i,j)}, \mathsf{r}_1^{(i,j)}, \mathsf{r}_\pi^{(i,j)}) \leftarrow \mathsf{PPRF.Eval}(\mathsf{k}, (\overline{\mathsf{T}}, (i,j)))$
$\forall (i,j) \in \mathbb{N}^2 : \mathsf{node}^{(i,j)} \in \mathsf{T}$,
    $\mathsf{CT}_0^{(i,j)} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_0, \mathsf{node}^{(i,j)}; \mathsf{r}_0^{(i,j)})$
    $\mathsf{CT}_1^{(i,j)} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_1, \mathsf{node}^{(i,j)}; \mathsf{r}_1^{(i,j)})$
    $\pi^{(i,j)} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{x}^{(i,j)}, (\mathsf{node}^{(i,j)}, \mathsf{r}_0^{(i,j)}, \mathsf{r}_1^{(i,j)}); \mathsf{r}_\pi^{(i,j)})$
```

7. Finally, output the updated (encrypted) tree $\overline{\mathsf{T}}$, the address for next iteration and possibly the outcome of the token.

```
return $(\overline{\mathsf{T}}, \mathsf{addr}, \mathsf{y})$
```

Figure 7.3: Specification of circuit $\widehat{\delta}$, as part of our updatable functional encryption scheme.

$(\mathsf{tag}_0, \mathsf{tag}_1)$ are independently sampled random tags. In this game, $\mathsf{CT}_1^{(i,j)}$ in the challenge ciphertext encrypts $\mathsf{node}_1^{(i,j)}$; the NIZK proofs are still simulated. This transition is negligible down to the IND-CPA security of PKE.

Game$_3$: In this game we hardwire a pre-computed lookup table to each circuit $\widehat{\delta_l}$, containing fixed inputs/outputs that allow to bypass the steps described in Figure 7.3. If the input to the circuit is on the lookup table, it will immediately return the corresponding output. The lookup tables are computed such that executing the tokens in sequence starting on the challenge ciphertext will propagate the execution over D_0 in the left branch and D_1 in the right branch. Because the challenge ciphertext evolves over time as tokens are executed, to argue this game hop we must proceed by hardwiring one input/output at the time, as follows: (1) We hardwire the input/output of the regular execution [iO property of Obf]; (2) we puncture the PPRF key of $\widehat{\delta_l}$ on the new hardwired input [functionality preservation under puncturing of PPRF + iO property of Obf]; (3) we replace the pseudorandom coins used to produce the hardwired output with real random coins [pseudorandomness at punctured points of PPRF]; (4) we use simulated NIZK proofs in the new hardwired output [zero-knowledge property of NIZK]; (5) we compute circuit δ_l independently on the right branch before encrypting the hardwired output [IND-CPA security of PKE].

Game$_4$: In all circuits $\widehat{\delta_l}$, we switch the decryption key sk_0 with sk_1 and perform the operations based on the right branch, i.e. we modify the circuits such that $\mathsf{node}^{(i,j)} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_1, \mathsf{CT}_1^{(i,j)})$. This hop can be upper-bounded by the distributional indistinguishability of Obf. To show this, we construct an adversary $(\mathcal{S}, \mathcal{B})$ against the DI game that runs adversary $\mathcal{A}$ as follows.

Sampler $\mathcal{S}$ runs $\mathcal{A}_0$ to get the challenge messages $(\mathsf{D}_0, \mathsf{D}_1)$ and circuits δ_l. Then, it produces the challenge ciphertext (same rules apply on Game$_3$ and Game$_4$), and compute circuits $\widehat{\delta_l}$ according to rules of Game$_3$ (with decryption key sk_0) on one hand and according to rules of Game$_4$ (with decryption key sk_1) on the other. Finally, it outputs the two vectors of circuits and the challenge ciphertext as auxiliary information.

Adversary $\mathcal{B}$ receives the obfuscated circuits $\overline{\delta}_l$ either containing sk_0 or sk_1 and the challenge ciphertext. With those, it runs adversary $\mathcal{A}_1$ perfectly simulating Game_3 or Game_4. $\mathcal{B}$ outputs whatever $\mathcal{A}_1$ outputs.

It remains to show that sampler $\mathcal{S}$ is *computationally* unpredictable. Suppose there is a predictor Pred that finds a differing input for the circuits output by sampler $\mathcal{S}$. It must be because either the output contains a NIZK proof for a false statement (which contradicts the soundness property of NIZK), or there is a collision in the Merkle tree (which contradicts the collision-resistance of H), or the predictor was able to guess the random tag in one of the ciphertexts (which contradicts the IND-CCA security of PKE). Note that (1) the random tag is high-entropy, so lucky guesses can be discarded; (2) we cannot rely only on IND-CPA security of PKE because we need the decryption oracle to check which random tag the predictor was able to guess to win the indistinguishability game against PKE. We also rely on the fact that adversary $\mathcal{A}_0$ is legitimate in its own game, so the outputs in clear of the tokens are the same in Game_3 and Game_4.

Game_5 : In this game, we remove the lookup tables introduced in Game_3. We remove one input/output at the time, from the last input/output pair added to the first, following the reverse strategy of that introduced in Game_3.

Game_6 : Here, the challenge ciphertext is computed exclusively from D_1 (with the same random tag on both branches). This transition is negligible down to the IND-CPA security of PKE.

Game_7 : In this game, we move back to regular (non-simulated) NIZK proofs in the challenge ciphertext. The distance to the previous game can be bounded by the zero-knowledge property of NIZK.

Game_8 : We now switch back the decryption key to sk_0 and perform the decryption operation on the left branch. Since NIZK is statistically sound, the circuits are functionally equivalent. We move from sk_1 to sk_0 one token at the time. This transition is down to the iO property of Obf. This game is identical to the real SEL game when the challenge bit $b = 1$, which concludes our proof.

$\square$

It is easy to check that the proposed scheme meets the correctness and efficiency properties as we defined in Section 7.2 for our primitive. The size of the ciphertext is proportional to the size of the cleartext. The size expansion of the token is however proportional to the number of steps of its execution, as the circuit $\overline{\delta}$ must be appropriately padded for the security proof.

Chapter 8

Conclusion and Open Problems

Function privacy is limited to what cannot be learned through trivial function evaluation. In this thesis we propose a precise definition of private functional encryption that models not only the privacy of encrypted messages but also that of (possibly) correlated tokens. Compared to previous definitions, ours is more general in that it can be used to model any function class through our unpredictability framework, capturing a strong degree of security in real world usage of functional encryption schemes, without the caveat of having a definition that is so strong that we stumble into the known impossibility results.

Several constructions achieving various degrees of security depending on the expressiveness of the supported function class and on the underlying assumptions have been proposed, including a concrete construction of an IBE scheme from groups of composite order. We leave, however, some open problems for future work, namely the construction of functional encryption schemes that achieve full PRIV security from simple forms of obfuscation or are more efficient under restricted versions of the PRIV model.

For general circuits, a possible path towards a solution to this open problem would be to consider the FE construction of Garg et al. [GGH$^+$13]. There, a token for a circuit C is (roughly speaking) an indistinguishability obfuscation of the circuit $\mathsf{C}(\mathsf{PKE.Dec}(\mathsf{sk}, \cdot))$. A natural question is whether this construction already achieves some form of privacy under the conjecture that the indistinguishability obfuscator achieves VGB obfuscation [BCKP14, Section 1.1]. For specific classes, one can follow the various constructions presented here and explore variations and optimizations of their underlying primitives. Indeed, since Res-PRIV constitutes a very mild weakening of PRIV, it could be that a modification of it allows the proof of security to be extended to the PRIV model.

Alternatively, a different direction would be to try to instantiate our DOX construction, which offers very strong privacy guarantees for the keyword search functionality, with an efficient FE scheme supporting the necessary function class. Currently we only know how to instantiate such construction from general-purpose FE schemes, such that in [GGH+13, Wat15, GGHZ14]. However, since we only require one exponentiation and one equality test, more efficient FE schemes for this particular functionality are likely to exist.

We also proposed the notion of *updatable functional encryption*, a new primitive that models functions as RAM programs, instead of circuits, and allows memory to be persistent across the execution of different programs. The problem at hand showed up to be quite challenging to realize, even when taking strong cryptographic primitives as building blocks. Still, one might wish to strengthen the proposed security model by allowing the adversary to obtain tokens adaptively, or by relaxing the legitimacy condition that imposes equal access patterns of extracted programs on left and right challenge messages using known results on Oblivious RAM. In this regard, we view our construction as a starting point towards the realization of other updatable functional encryption schemes.

Bibliography

[AAB$^+$15] Shashank Agrawal, Shweta Agrawal, Saikrishna Badrinarayanan, Abishek Kumarasubramanian, Manoj Prabhakaran, and Amit Sahai. On the practical security of inner product functional encryption. In *Public-Key Cryptography – PKC 2015*, volume 9020 of *LNCS*, pages 777–798. Springer, 2015.

[ABC$^+$08] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *Journal of Cryptology*, 21(3):350–391, 2008.

[ABDP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In *Public-Key Cryptography – PKC 2015*, volume 9020 of *LNCS*, pages 733–751. Springer, 2015.

[ABDP16] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Better security for functional encryption for inner product evaluations. IACR Cryptology ePrint Archive, Report 2016/011, 2016. `http://eprint.iacr.org/2016/011`.

[ABF16] Afonso Arriaga, Manuel Barbosa, and Pooya Farshim. Private functional encryption: Indistinguishability-based definitions and constructions from obfuscation. In *Progress in Cryptology – INDOCRYPT 2016*, volume 10095 of *LNCS*, pages 227–247. Springer, 2016.

[ABG$^+$13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. IACR Cryptology ePrint Archive, Report 2013/689, 2013. http://eprint.iacr.org/2013/689.

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In *Advances in Cryptology – CRYPTO 2015*, volume 9216 of *LNCS*, pages 657–677. Springer, 2015.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 21–40. Springer, 2011.

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *Advances in Cryptology – CRYPTO 2013*, volume 8043 of *LNCS*, pages 500–518. Springer, 2013.

[AIT16] Afonso Arriaga, Vincenzo Iovino, and Qiang Tang. Updatable functional encryption. In *Paradigm-shifting Crypto – MYCRYPT 2016*, LNCS. Springer, 2016.

[AS16] Prabhanjan Ananth and Amit Sahai. Functional encryption for Turing machines. In *Theory of Cryptography – TCC 2016-A*, volume 9562 of *LNCS*, pages 125–153. Springer, 2016.

[ATR14] Afonso Arriaga, Qiang Tang, and Peter Ryan. Trapdoor privacy in asymmetric searchable encryption schemes. In *Progress in Cryptology – AFRICACRYPT 2014*, volume 8469 of *LNCS*, pages 31–50. Springer, 2014.

[BBC$^+$14] Boaz Barak, Nir Bitansky, Ran Canetti, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Obfuscation for evasive functions. In *Theory of Cryptography – TCC 2014*, volume 8349 of *LNCS*, pages 26–51. Springer, 2014.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in*

Cryptology – EUROCRYPT 2000, volume 1807 of *LNCS*, pages 259–274. Springer, 2000.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[BC14] Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. *Journal of Cryptology*, 27(2):317–357, 2014.

[BCKP14] Nir Bitansky, Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On virtual grey box obfuscation for general circuits. In *Advances in Cryptology – CRYPTO 2014*, volume 8617 of *LNCS*, pages 108–125. Springer, 2014.

[BDOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522. Springer, 2004.

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.

[BF13] Manuel Barbosa and Pooya Farshim. On the semantic security of functional encryption schemes. In *Public-Key Cryptography – PKC 2013*, volume 7778 of *LNCS*, pages 143–161. Springer, 2013.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, 2001.

[BGK$^+$14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 221–238. Springer, 2014.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *ACM Conference on Computer and Communications Security – CCS 2012*, pages 784–796. ACM, 2012.

[BM14] Christina Brzuska and Arno Mittelbach. Indistinguishability obfuscation versus multi-bit point obfuscation with auxiliary input. In *Advances in Cryptology – ASIACRYPT 2014*, volume 8874 of *LNCS*, pages 142–161. Springer, 2014.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409—426. Springer, 2006.

[BR14a] Zvika Brakerski and Guy N. Rothblum. Black-box obfuscation for d-CNFs. In *ACM Conference on Innovations in Theoretical Computer Science – ITCS 2015*, pages 235–250. ACM, 2014.

[BR14b] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *Theory of Cryptography – TCC 2014*, volume 8349 of *LNCS*, pages 1–25. Springer, 2014.

[BRS13a] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In *Advances in Cryptology – CRYPTO 2013*, volume 8043 of *LNCS*, pages 461–478. Springer, 2013.

[BRS13b] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private subspace-membership encryption and its applications. In *Advances in Cryptology – ASIACRYPT 2013*, volume 8269 of *LNCS*, pages 255–275. Springer, 2013.

[BST14] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In *Advances in Cryptology – ASIACRYPT 2014*, volume 8874 of *LNCS*, pages 102–121. Springer, 2014.

[BST16] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Contention in cryptoland: Obfuscation, leakage and UCE. In *Theory of Cryptography – TCC 2016-A*, volume 9563 of *LNCS*, pages 542–564. Springer, 2016.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography – TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, 2011.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *LNCS*, pages 290–307. Springer, 2006.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography Conference – TCC 2007*, volume 4392 of *LNCS*, pages 535–554. Springer, 2007.

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *Advances in Cryptology – CRYPTO 1997*, volume 1294 of *LNCS*, pages 455–469. Springer, 1997.

[CCHR15] Ran Canetti, Yilei Chen, Justin Holmgren, and Mariana Raykova. Succinct adaptive garbled RAM. IACR Cryptology ePrint Archive, Report 2015/1074, 2015. `http://eprint.iacr.org/2015/1074`.

[CH16] Ran Canetti and Justin Holmgren. Fully succinct garbled RAM. In *ACM Conference on Innovations in Theoretical Computer Science – ITCS 2016*, pages 169–178. ACM, 2016.

[CHJV14] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Indistinguishability obfuscation of iterated circuits and RAM programs. IACR Cryptology ePrint Archive, Report 2014/769, 2014. `http://eprint.iacr.org/2014/769`.

[CHJV15] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In

ACM on Symposium on Theory of Computing – STOC 2015, pages 429–437. ACM, 2015.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 360–363. Springer, 2001.

[CRV10] Ran Canetti, Guy N. Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In *Theory of Cryptography – TCC 2010*, volume 5978 of *LNCS*, pages 72–89. Springer, 2010.

[CV13] Ran Canetti and Vinod Vaikuntanathan. Obfuscating branching programs using black-box pseudo-free groups. IACR Cryptology ePrint Archive, Report 2013/500, 2013. http://eprint.iacr.org/2013/500.

[DIJ+13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In *Advances in Cryptology – CRYPTO 2013*, volume 8043 of *LNCS*, pages 519–535. Springer, 2013.

[DIPV14] Yvo Desmedt, Vincenzo Iovino, Giuseppe Persiano, and Ivan Visconti. Controlled homomorphic encryption: Definition and construction. IACR Cryptology ePrint Archive, Report 2014/989, 2014. http://eprint.iacr.org/2014/989.

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

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure mpc from indistinguishability obfuscation. In *Theory of Cryptography Conference – TCC 2014*, volume 8349 of *LNCS*, pages 74–94. Springer, 2014.

[GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure attribute based encryption from multilinear maps. IACR Cryptology ePrint Archive, Report 2014/622, 2014. http://eprint.iacr.org/2014/622.

[GHL$^+$14] Craig Gentry, Shai Halevi, Steve Lu, Rafail Ostrovsky, Mariana Raykova, and Daniel Wichs. Garbled RAM revisited. In *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 405–422. Springer, 2014.

[GHRW14] Craig Gentry, Shai Halevi, Mariana Raykova, and Daniel Wichs. Outsourcing private RAM computation. In *IEEE Symposium on Foundations of Computer Science – FOCS 2014*, pages 404–413. IEEE Computer Society, 2014.

[GJKS15] Vipul Goyal, Abhishek Jain, Venkata Koppula, and Amit Sahai. Functional encryption for randomized functionalities. In *Theory of Cryptography – TCC 2015*, volume 9015 of *LNCS*, pages 325–351. Springer, 2015.

[GK05] S. Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *IEEE Symposium on Foundations of Computer Science – FOCS 2005*, pages 553–562. IEEE Computer Society, 2005.

[GKP$^+$13] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *ACM Symposium on Theory of Computing – STOC 2013*, pages 555–564. ACM, 2013.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.

[GR14] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. *Journal of Cryptology*, 27(3):480–505, 2014.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, 2008.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, 2012.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In *Advances in Cryptology – CRYPTO 2015*, volume 9216 of *LNCS*, pages 503–523. Springer, 2015.

[IPS15] Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. In *Theory of Cryptography – TCC 2015*, volume 9015 of *LNCS*, pages 668–697. Springer, 2015.

[KSW13] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *Journal of Cryptology*, 26(2):191–224, 2013.

[LO13] Steve Lu and Rafail Ostrovsky. How to garble RAM programs? In *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 719–734. Springer, 2013.

[LOS$^+$10] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, 2010.

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *ACM Symposium on Theory of Computing – STOC 1990*, pages 427–437. ACM, 1990.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. IACR Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/2010/556.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – CRYPTO 1984*, volume 196 of *LNCS*, pages 47–53. Springer, 1984.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. In *ACM Symposium on Theory of Computing – STOC 2014*, pages 475–484. ACM, 2014.

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In *Advances in Cryptology – CRYPTO 2015*, volume 9216 of *LNCS*, pages 678–697. Springer, 2015.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Symposium on Foundations of Computer Science – SFCS 1986*, pages 162–167. IEEE Computer Society, 1986.

Appendix A

Detailed Security Proof of DOX Transform

Theorem 1 (Res-PRIV security of DOX)**.** *If* FE *is an* IND-CPA *secure functional encryption scheme,* PRP *is pseudorandom and* Obf *is a* DI-*secure obfuscator then scheme* $\mathsf{DOX}[\mathsf{FE}, \mathsf{PRP}, \mathsf{Obf}]$ *is* Res-PRIV *secure. More precisely, for any adversary* $(\mathcal{S}, \mathcal{A})$ *in game* Res-PRIV *against scheme* $\mathsf{DOX}[\mathsf{FE}, \mathsf{PRP}, \mathsf{Obf}]$*, in which* $\mathcal{A}$ *places at most* q *queries to* TGen *oracle and* $\mathcal{S}$ *outputs a tuple* $(\mathbf{w}_0, \mathbf{w}_1, \mathbf{m}_0, \mathbf{m}_1, z)$ *such that* $|\mathbf{w}_0| = |\mathbf{w}_1| = t$ *and* $|\mathbf{m}_0| = |\mathbf{m}_1| = s$*, there exists adversaries* $\mathcal{B}_1$*,* $\mathcal{B}_2$*,* $\mathcal{B}_3$*,* $(\mathcal{S}_4, \mathcal{B}_4)$*,* $\mathcal{B}_5$*,* $(\mathcal{S}_6, \mathcal{B}_6)$ *such that*

$$\begin{aligned}\mathbf{Adv}^{\text{res-priv}}_{\mathsf{DOX},\mathcal{S},\mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\mathsf{prp}}_{\mathsf{PRP},\mathcal{B}_1}(\lambda) + 2 \cdot (t + s + q) \cdot \mathbf{Adv}^{\text{ow}}_{\mathsf{Obf},\mathcal{B}_2}(\lambda) + \\ 2 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{FE},\mathcal{B}_3}(\lambda) + 2 \cdot \mathbf{Adv}^{\text{di}}_{\mathsf{Obf},\mathcal{S}_4,\mathcal{B}_4}(\lambda) + 2 \cdot (\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{FE},\mathcal{B}_5}(\lambda) + \\ \frac{s \cdot q}{|\mathsf{WSp}_\lambda|}) + \mathbf{Adv}^{\text{di}}_{\mathsf{Obf},\mathcal{S}_6,\mathcal{B}_6}(\lambda) \ .\end{aligned}$$

Proof. The proof follows from a sequence of six game hops. We refer the reader to Figure A.4 and Figure A.5 for a formal description of each game in a code-based language. Since the definition of Res-PRIV composes for stateless samplers, we assume $\mathcal{A}$ calls LR oracle exactly once.

Game$_0$: This game is identical to the Res-PRIV game.

Game$_1$: Instead of a PRP, a truly random permutation (simulated via lazy sampling) is used in token generation. The table used to maintain the lazy sampling, which we denote by T, has at most $(t+q)$ entries. The distance to the previous game can be bounded using the security of the PRP.

Game$_2$: When sampler $\mathcal{S}$ outputs, we generate PRP values of all messages in $\mathbf{m}_b$ as well. Since Game$_1$, these are now simulated via lazy sampling, which causes the expansion of table T to at most $(t+s+q)$ entries. All keywords and messages whose PRP value was generated are registered in list. Before setting the outcome of the game, if there are values w_1 and w_2 in list such that $\mathsf{w}_1 = T[\mathsf{w}_2]$, game aborts. Throughout the game $T[\mathsf{w}]$ is obfuscated, so the distance to the previous game can be upper bounded by the one-wayness property of the obfuscator.

Game$_3$: LR oracle computes the vector of ciphertexts $\mathbf{c}$ by encrypting $T[\mathbf{m}_b]$ instead of $\mathbf{m}_b$. The distance to the previous game can be upper bounded using the IND-CPA security property of the underlying FE scheme.

Game$_4$: We say a keyword w is *unpaired* if $\mathsf{w} \in \mathbf{w}_b$ and $\mathsf{w} \notin \mathbf{m}_b$. All first-phase queries to TGen oracle are recorded in $\mathsf{FirstPhase}$ list, i.e. all keywords $\mathcal{A}$ queries to TGen oracle before calling LR. During the simulation of LR oracle and second-phase TGen oracle, if w is an unpaired keyword not in $\mathsf{FirstPhase}$ list, we extract its token from circuit $(\mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}]) \vee \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{r}]))$, where r is a *fresh* random value uniformly sampled from WSp_λ. We precise that by *fresh* we mean that a new and independent random value is sampled each time, even in case of multiple token extractions of the same keyword. The distance to the previous game can be upper bounded to the DI security of the obfuscator.

Game$_5$: Analogously, we say a message m is *unpaired* if $\mathsf{m} \in \mathbf{m}_b$ and $\mathsf{m} \notin \mathbf{w}_b$. During the simulation of LR oracle, if m is an unpaired message not in $\mathsf{FirstPhase}$ list, we encrypt r instead of $T[\mathsf{m}]$, where r is a *fresh* random value uniformly sampled from WSp_λ. We precise that by *fresh* we mean that a new and independent random value is sampled each time, even in case of repetitions of the same message. We bound this hop down to IND-CPA.

Game$_6$: In this game, irrespective of the bit b, we use the second set of keywords for challenge token generation.

We now analyze the transitions between each game and the reduction of Game$_5$ to DI game.

Game$_0$ TO Game$_1$. Any adversary $(\mathcal{S}, \mathcal{A})$ with visible advantage difference in these two games can be converted to an adversary $\mathcal{B}_1$ against the security of PRP. Assume that lazy sampling is implemented using a table T, i.e., $T[\mathsf{w}]$ indicates the random value assigned to w. Algorithm $\mathcal{B}_1$ runs adversary $(\mathcal{S}, \mathcal{A})$ inside it, simulating all the details of Game$_0$, bar the computation of the PRP. For this, algorithm $\mathcal{B}_1$ uses its FN oracle. When $\mathcal{A}$ terminates, $\mathcal{B}_1$ checks if $\mathcal{A}$ succeeded in winning the game. If so, it outputs 0. Otherwise, it outputs 1.

When the FN oracle implements the PRP, Game$_0$ is simulated for $\mathcal{A}$, and when FN implements a random permutation, Game$_1$ is simulated. Therefore,

$$\Pr[\text{Game}_0(1^\lambda)] - \Pr[\text{Game}_1(1^\lambda)] = \mathbf{Adv}^{\mathsf{prp}}_{\mathsf{PRP},\mathcal{B}_1}(\lambda) \ .$$

Game$_1$ TO Game$_2$. Both games are exactly the same unless the bad event that causes abortion is triggered. Game$_2$ aborts if there are values w_1 and w_2 in list such that $\mathsf{w}_1 = T[\mathsf{w}_2]$. We show that an adversary $(\mathcal{S}, \mathcal{A})$ that triggers the bad event in Game$_2$ can be converted to an adversary $\mathcal{B}_2$ against the one-wayness property of Obf. For an intuition on this game hop, observe that all occurrences of $T[\mathsf{w}]$ are obfuscated and $T[\mathsf{w}]$ is uniformly distributed.

During the simulation of Game$_2$, table T expands up to $(t+s+q)$ entries. Algorithm $\mathcal{B}_2$ receives in its challenge $(t+q)$ obfuscated copies of a random point circuit. At the beginning of its execution, $\mathcal{B}_2$ randomly guesses the first occurrence of w_2 in the game, by sampling i is uniformly from $\{1, ..., (t+s+q)\}$. (Keyword w_2 is of course unknown to $\mathcal{B}_2$ at this point, the guess reflects a prediction of when such keyword involved in the bad event will be added to table T.) Then, $\mathcal{B}_2$ simulates for adversary $(\mathcal{S}, \mathcal{A})$ all the details of Game$_2$ until a new keyword comes that will cause table T to expand to i entries. Instead of sampling $T[\mathsf{w}_2]$, $\mathcal{B}_2$ embeds one of its challenge circuits in the computation of $\mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}_2]])$. (If w_2 is a message, nothing needs to be done.) Thenceforth, $\mathcal{B}_2$ embeds a new circuit from its challenge each time it needs to extract a token for w_2. In any case, $\mathcal{B}_2$ never needs more than $(t+q)$ challenge circuits to complete its simulation.

At the end of the game, if $\mathcal{B}_2$'s guess is correct, which happens with probability

$1/(t+s+q)$, there is $\mathsf{w}_1 \in \mathsf{list}$ such that $\mathsf{w}_1 = T[\mathsf{w}_2]$. This equality can be checked by evaluating w_1 on one of $\mathcal{B}_2$'s obfuscated circuits. If so, $\mathcal{B}_2$ outputs w_1 and wins the game. Hence,

$$\Pr[\mathrm{Game}_1(1^\lambda)] - \Pr[\mathrm{Game}_2(1^\lambda)] \leq (t+s+q) \cdot \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{Obf},\mathcal{B}_2}(\lambda) \ .$$

Game$_2$ TO Game$_3$. Any legitimate adversary $(\mathcal{S}, \mathcal{A})$ with visible advantage difference in these two games can be converted to an adversary $\mathcal{B}_3$ against IND-CPA security of FE. For an intuition on this reduction, observe that all tokens are extracted from circuits of the form $(\mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}]) \vee \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]]))$, which return 1 when evaluated on both w and $T[\mathsf{w}]$. Illegitimate tokens that would allow to distinguish encryptions of m from encryption of $T[\mathsf{m}]$ have been excluded in Game$_2$, given that the game aborts if adversary $(\mathcal{S}, \mathcal{A})$ outputs a value sampled for the simulation of the random permutation.

Algorithm $\mathcal{B}_3$ runs adversary $(\mathcal{S}, \mathcal{A})$ inside it, simulating all the details common to Game$_2$ and Game$_3$. $\mathcal{B}_3$ receives mpk and runs adversary $\mathcal{A}$ with it. For token-generation and encryption, $\mathcal{B}_3$ relies on its oracles. When $\mathcal{B}_3$ needs to compute a token for some keyword w, it queries its own TGEN oracle with circuit $(\mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}]) \vee \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]]))$. When $\mathcal{B}_3$ needs to compute the encryption of some message m in Game$_2$ or $T[\mathsf{m}]$ in Game$_3$, it queries its own LR oracle with $(\mathsf{m}, T[\mathsf{m}])$. The ciphertexts output by LR oracle in game IND-CPA allow $\mathcal{B}_3$ to interpolate between the simulation of Game$_2$ and Game$_3$. The simulation is perfect. Eventually, $\mathcal{A}$ outputs b', which $\mathcal{B}_3$ forwards as its own guess.

Now, let's analyze legitimacy of $\mathcal{B}_3$. Legitimacy condition of IND-CPA requires that for all C queried to TGEN and all $(\mathsf{m}_0, \mathsf{m}_1)$ queried to LR, we have that $\mathsf{C}(\mathsf{m}_0) = \mathsf{C}(\mathsf{m}_1)$. In the execution of $\mathcal{B}_3$, queried circuits are of the form $(\mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}]) \vee \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]]))$ and queried messages of the form $(\mathsf{m}, T[\mathsf{m}])$. More precisely, legitimacy requires that

$\forall \mathsf{w} \in \mathsf{TList}, \forall \mathsf{m} \in \mathsf{MList},$

$$\begin{aligned}
(\mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}]) \vee \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]]))(\mathsf{m}) &= (\mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}]) \vee \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]]))(T[\mathsf{m}]) \\
\Leftrightarrow (\mathsf{C}[\mathsf{w}] \vee \mathsf{C}[T[\mathsf{w}]])(\mathsf{m}) &= (\mathsf{C}[\mathsf{w}] \vee \mathsf{C}[T[\mathsf{w}]])(T[\mathsf{m}]) \quad /\!/ \text{ func. preserving} \\
\Leftrightarrow \mathsf{C}[\mathsf{w}](\mathsf{m}) \vee \mathsf{C}[T[\mathsf{w}]](\mathsf{m}) &= \mathsf{C}[\mathsf{w}](T[\mathsf{m}]) \vee \mathsf{C}[T[\mathsf{w}]](T[\mathsf{m}]) \\
\Leftrightarrow \mathsf{C}[\mathsf{w}](\mathsf{m}) &= \mathsf{C}[T[\mathsf{w}]](T[\mathsf{m}]) \quad /\!/ \text{ bad event in Game}_2 \\
\Leftrightarrow (\mathsf{w} \overset{?}{=} \mathsf{m}) &= (T[\mathsf{w}] \overset{?}{=} T[\mathsf{m}]) \\
\Leftrightarrow & \mathsf{True} \ .
\end{aligned}$$

Therefore, $\mathcal{B}_3$ is a legitimate adversary against IND-CPA and we have that

$$\Pr[\text{Game}_2(1^\lambda)] - \Pr[\text{Game}_3(1^\lambda)] = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{FE},\mathcal{B}_3}(\lambda) \ .$$

Game$_3$ TO Game$_4$. Any legitimate adversary $(\mathcal{S}, \mathcal{A})$ with visible advantage difference in these two games can be converted to an adversary $(\mathcal{S}_4, \mathcal{B}_4)$ against DI security of Obf. The intuition here is the following: Without a ciphertext that encrypts $T[\mathsf{w}]$, the adversary cannot detect if tokens for w are extracted from $(\mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}]) \vee \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]]))$ or from $(\mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}]) \vee \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{r}]))$, where r is a fresh random value uniformly sampled from WSp_λ. Details of adversary $(\mathcal{S}_4, \mathcal{B}_4)$ are shown in Figure A.1.

Sampler $\mathcal{S}_4$ computes two $t \times (t+q)$ matrices $\mathbf{M}_0$ and $\mathbf{M}_1$. Each row in $\mathbf{M}_0$ contains $(t+q)$ repetitions of a *unique* random point circuit. $\mathbf{M}_1$ contains $t \times (t+q)$ fresh random point circuits. $\mathcal{S}_4$ is clearly unpredictable. Algorithm $\mathcal{B}_4$ runs $\mathcal{S}$ and $\mathcal{A}$ inside it, simulating all the details common to Game$_3$ and Game$_4$, which only differ on unpaired keywords not in $\mathsf{FirstPhase}$ list. For those, $\mathcal{B}_4$ carefully picks circuits from its challenge matrix of obfuscated circuits: A new row is assigned to a new keyword; a circuit is picked from a new column in case of repetitions. If $\mathbf{M}_0$ is selected in game DI, algorithm $\mathcal{B}_4$ will simulate Game$_3$. On the other hand, if $\mathbf{M}_1$ is selected in game DI, algorithm $\mathcal{B}_4$ will simulate Game$_4$. Finally, when $\mathcal{A}$ outputs its guess, $\mathcal{B}_4$ checks if $\mathcal{A}$ succeeded in winning the game. If so, $\mathcal{B}_4$ outputs 0. Otherwise, it outputs 1. Therefore, we have that

$$\Pr[\text{Game}_3(1^\lambda)] - \Pr[\text{Game}_4(1^\lambda)] = \mathbf{Adv}^{\text{di}}_{\mathsf{Obf},\mathcal{S}_4,\mathcal{B}_4}(\lambda) \ .$$

```
$\underline{\mathcal{S}_4(1^\lambda, st)}$:
for $i \in \{1, \dots, t\}$
    $T[i] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
    for $j \in \{1, \dots, (t+q)\}$
        $\mathbf{M}_0[i][j] \leftarrow \mathsf{C}[T[i]]$
        $\mathsf{r} \leftarrow\$ \mathsf{WSp}_\lambda$
        $\mathbf{M}_1[i][j] \leftarrow \mathsf{C}[\mathsf{r}]$
return $(\mathbf{M}_0, \mathbf{M}_1, \epsilon)$
```

- -

```
$\underline{\mathcal{B}_4(1^\lambda)}$:
$(\mathsf{msk}, \mathsf{mpk}) \leftarrow\$ \mathsf{FE.Gen}(1^\lambda)$
$b \leftarrow\$ \{0,1\}$
$\overline{\mathbf{M}} \leftarrow\$ \mathrm{DI.SAM}(\epsilon)$
$b' \leftarrow\$ \mathcal{A}^{\mathrm{LR},\mathrm{TGEN}}(\mathsf{mpk})$
return $(b = b')$

$\underline{\mathrm{TGEN}(\mathsf{w})}$:
$\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
if $T[\mathsf{w}] = \perp$ then
    $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
    $\mathsf{List} \leftarrow \mathsf{List} : \mathsf{w}$
if $(\mathsf{w} \in \mathbf{w}_b \wedge \mathsf{w} \notin \mathbf{m}_b \wedge \mathsf{w} \notin \mathsf{FirstPhase})$
    $\overline{\mathsf{C}}_\mathsf{R} \leftarrow \mathsf{Select}(\mathsf{w})$
else
    $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
$\mathsf{tk} \leftarrow\$ \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $\mathsf{tk}$
```

```
$\underline{\mathrm{LR}(st)}$:
$\mathsf{FirstPhase} \leftarrow \mathsf{List}$
$(\mathbf{w}_0, \mathbf{w}_1, \mathbf{m}_0, \mathbf{m}_1, z) \leftarrow\$ \mathcal{S}(st)$
for all $\mathsf{w} \in \mathbf{w}_b \cup \mathbf{m}_b$
    if $T[\mathsf{w}] = \perp$ then
        $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
        $\mathsf{List} \leftarrow \mathsf{List} : \mathsf{w}$
$\mathbf{c} \leftarrow\$ \mathsf{FE.Enc}(\mathsf{mpk}, T[\mathbf{m}_b])$
for all $\mathsf{w} \in \mathbf{w}_b$
    $\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
    if $(\mathsf{w} \notin \mathbf{m}_b \wedge \mathsf{w} \notin \mathsf{FirstPhase})$ then
        $\overline{\mathsf{C}}_\mathsf{R} \leftarrow \mathsf{Select}(\mathsf{w})$
else
    $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
    $\mathbf{tk} \leftarrow \mathbf{tk} : \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $(\mathbf{tk}, \mathbf{c}, z)$
```

```
$\underline{\mathsf{Select}(\mathsf{w})}$ :
if $(\mathsf{Row}[\mathsf{w}] = \perp)$ then
    // assign next available row to w
    $\mathsf{Row}[\mathsf{w}] \leftarrow |\mathsf{Row}| + 1$
    // initialize counter for w
    $\mathsf{Column}[\mathsf{Row}[\mathsf{w}]] \leftarrow 0$
// increase counter for w
$\mathsf{Column}[\mathsf{Row}[\mathsf{w}]] \leftarrow \mathsf{Column}[\mathsf{Row}[\mathsf{w}]] + 1$
// return fresh obfuscated circuit
return $(\overline{\mathbf{M}}[\mathsf{Row}[\mathsf{w}]][\mathsf{Column}[\mathsf{Row}[\mathsf{w}]]])$
```

Figure A.1: DI adversary $(\mathcal{S}_4, \mathcal{B}_4)$, as part of proof of Theorem 8.

Game$_4$ TO Game$_5$. Any legitimate adversary $(\mathcal{S}, \mathcal{A})$ with visible advantage difference in these two games can be converted to an adversary $\mathcal{B}_5$ against IND-CPA security of FE. The intuition here is simple: Without a token for m, the adversary cannot detect if we encrypt a fresh random value r instead if $T[\mathsf{m}]$. Details of adversary $\mathcal{B}_5$ are shown in Figure A.2.

Let d denote the challenge bit in the IND-CPA game for FE. Let d' denote $\mathcal{B}_5$'s output. By definition, we have that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{FE},\mathcal{B}_5}(\lambda) = \Pr[d' = 0 | d = 0] - \Pr[d' = 0 | d = 1].$$

Also, let Ter be the event that $\mathcal{B}_5$ terminates because $\mathcal{A}$ queried $\mathsf{w} \in \mathsf{RList}$. We have that

$$\begin{aligned}
\Pr[d' = 0 | d = 0] &= \Pr[d' = 0 | d = 0 \wedge \neg\mathsf{Ter}] \cdot \Pr[\neg\mathsf{Ter}] + \Pr[d' = 0 | d = 0 \wedge \mathsf{Ter}] \cdot \Pr[\mathsf{Ter}] \\
&= \Pr[d' = 0 | d = 0 \wedge \neg\mathsf{Ter}] \cdot \Pr[\neg\mathsf{Ter}] + 0 \quad /\!/ \ \mathcal{B}_5 \text{ never outputs 0 if } \mathsf{Ter} \\
&= \Pr[d' = 0 | d = 0 \wedge \neg\mathsf{Ter}] \cdot (1 - \frac{s \cdot q}{|\mathsf{WSp}_\lambda|}) \quad /\!/ \ d = 0, \text{ so } \mathsf{RList} \text{ is hidden} \\
&\geq \Pr[d' = 0 | d = 0 \wedge \neg\mathsf{Ter}] - \frac{s \cdot q}{|\mathsf{WSp}_\lambda|} = \Pr[\mathrm{Game}_4(1^\lambda)] - \frac{s \cdot q}{|\mathsf{WSp}_\lambda|}
\end{aligned}$$

$$\begin{aligned}
\Pr[d' = 0 | d = 1] &= \Pr[d' = 0 | d = 1 \wedge \neg\mathsf{Ter}] \cdot \Pr[\neg\mathsf{Ter}] + \Pr[d' = 0 | d = 1 \wedge \mathsf{Ter}] \cdot \Pr[\mathsf{Ter}] \\
&= \Pr[d' = 0 | d = 1 \wedge \neg\mathsf{Ter}] \cdot \Pr[\neg\mathsf{Ter}] + 0 \quad /\!/ \ \mathcal{B}_5 \text{ never outputs 0 if } \mathsf{Ter} \\
&\leq \Pr[d' = 0 | d = 1 \wedge \neg\mathsf{Ter}] = \Pr[\mathrm{Game}_5(1^\lambda)].
\end{aligned}$$

We now analyze legitimacy of $\mathcal{B}_5$. Legitimacy condition of IND-CPA requires that for all C queried to $\textsc{TGen}$ and all $(\mathsf{m}_0, \mathsf{m}_1)$ queried to LR, we have that $\mathsf{C}(\mathsf{m}_0) = \mathsf{C}(\mathsf{m}_1)$. In the execution of $\mathcal{B}_5$, queried circuits are of the form $(\mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}]) \vee \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]]))$ and queried messages of the form $(T[\mathsf{m}], \mathsf{r})$. More precisely, legitimacy requires that $\forall \mathsf{w} \in \mathsf{TList}, \forall \mathsf{m} \in \mathbf{m}_b$ s.t. $\mathsf{m} \notin \mathbf{w}_b \wedge \mathsf{m} \notin \mathsf{FirstPhase}, \forall \mathsf{r} \in \mathsf{RList}$, we have that

$$\begin{aligned}
(\mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}]) \vee \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]]))(T[\mathsf{m}]) &= (\mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}]) \vee \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]]))(\mathsf{r}) \\
\Leftrightarrow (\mathsf{C}[\mathsf{w}] \vee \mathsf{C}[T[\mathsf{w}]])(T[\mathsf{m}]) &= (\mathsf{C}[\mathsf{w}] \vee \mathsf{C}[T[\mathsf{w}]])(\mathsf{r}) \quad /\!/ \text{ func. preserving} \\
\Leftrightarrow \mathsf{C}[\mathsf{w}](T[\mathsf{m}]) \vee \mathsf{C}[T[\mathsf{w}]](T[\mathsf{m}]) &= \mathsf{C}[\mathsf{w}](\mathsf{r}) \vee \mathsf{C}[T[\mathsf{w}]](\mathsf{r}) \\
\Leftrightarrow \mathsf{C}[T[\mathsf{w}]](T[\mathsf{m}]) &= \mathsf{C}[\mathsf{w}](\mathsf{r}) \vee \mathsf{C}[T[\mathsf{w}]](\mathsf{r}) \quad /\!/ \text{ bad event in } \mathrm{Game}_2 \\
\Leftrightarrow \mathsf{C}[T[\mathsf{w}]](T[\mathsf{m}]) &= \mathsf{C}[T[\mathsf{w}]](\mathsf{r}) \quad /\!/ \ \mathcal{B}_5 \text{ outputs 1, } \mathsf{Ter} \text{ event} \\
\Leftrightarrow \mathsf{C}[\mathsf{w}](\mathsf{m}) &= \mathsf{C}[T[\mathsf{w}]](\mathsf{r}) \\
\Leftrightarrow 0 &= \mathsf{C}[T[\mathsf{w}]](\mathsf{r}) \quad /\!/ \text{ Res-PRIV restriction} \\
\Leftrightarrow 0 &= 0 \quad /\!/ \text{ with high probability } \mathsf{C}[T[\mathsf{w}]](\mathsf{r}) = 0 \ .
\end{aligned}$$

Therefore,

$$\Pr[\mathrm{Game}_4(1^\lambda)] - \Pr[\mathrm{Game}_5(1^\lambda)] \leq \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{FE},\mathcal{B}_5}(\lambda) + \frac{s \cdot q}{|\mathsf{WSp}_\lambda|}.$$

```
$\mathcal{B}_5(1^\lambda, \mathsf{mpk})$:
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\mathrm{LR},\mathrm{TGen}}(\mathsf{mpk})$
if ($\exists\ \mathsf{w}_1, \mathsf{w}_2 \in \mathsf{List}$ s.t. $\mathsf{w}_1 = T[\mathsf{w}_2]$) abort
return $(\neg(b = b'))$
```

```
$\mathrm{LR}(st)$:
$\mathsf{FirstPhase} \leftarrow \mathsf{List}$
$(\mathbf{w}_0, \mathbf{w}_1, \mathbf{m}_0, \mathbf{m}_1, z) \leftarrow\$ \mathcal{S}(st)$
for all $\mathsf{w} \in \mathbf{w}_b \cup \mathbf{m}_b$
  if $T[\mathsf{w}] = \perp$ then
    $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
    $\mathsf{List} \leftarrow \mathsf{List} : \mathsf{w}$
for all $\mathsf{m} \in \mathbf{m}_b$
  if ($\mathsf{m} \notin \mathbf{w}_b \wedge \mathsf{m} \notin \mathsf{FirstPhase}$) then
    $\mathsf{r} \leftarrow\$ \mathsf{WSp}_\lambda$
    $\mathsf{RList} \leftarrow \mathsf{RList} : \mathsf{r}$
    $\mathsf{c} \leftarrow\$ \text{IND-CPA.LR}(T[\mathsf{m}], \mathsf{r})$
  else
    $\mathsf{c} \leftarrow\$ \mathsf{FE.Enc}(\mathsf{mpk}, T[\mathsf{m}])$
  $\mathbf{c} \leftarrow \mathbf{c} : \mathsf{c}$
for all $\mathsf{w} \in \mathbf{w}_b$
  $\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
  if ($\mathsf{w} \notin \mathbf{m}_b \wedge \mathsf{w} \notin \mathsf{FirstPhase}$) then
    $\mathsf{r} \leftarrow\$ \mathsf{WSp}_\lambda$
    $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{r}])$
  else
    $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
  $\mathbf{tk} \leftarrow \mathbf{tk} : \text{IND-CPA.TGen}((\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $(\mathbf{tk}, \mathbf{c}, z)$
```

```
$\mathrm{TGen}(\mathsf{w})$:
if $\mathsf{w} \in \mathsf{RList}$ exit 1 // $\mathcal{B}_5$ terminates and outputs 1
$\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
if $T[\mathsf{w}] = \perp$ then
  $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
  $\mathsf{List} \leftarrow \mathsf{List} : \mathsf{w}$
if ($\mathsf{w} \in \mathbf{w}_b \wedge \mathsf{w} \notin \mathbf{m}_b \wedge \mathsf{w} \notin \mathsf{FirstPhase}$)
  $\mathsf{r} \leftarrow\$ \mathsf{WSp}_\lambda$
  $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{r}])$
else
  $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
$\mathsf{tk} \leftarrow\$ \text{IND-CPA.TGen}((\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $\mathsf{tk}$
```

Figure A.2: IND-CPA adversary $\mathcal{B}_5$, as part of proof of Theorem 8.

Game$_5$ TO Game$_6$. In this game, irrespective of the bit, we use the second set of keywords for challenge token generation. We construct an adversary $(\mathcal{S}_6, \mathcal{B}_6)$ against DI as follows. $\mathcal{B}_6$ generates by itself a master secret key and master public key pair $(\mathsf{msk}, \mathsf{mpk})$, then runs $\mathcal{A}(\mathsf{mpk})$. First-phase TGen queries are answered using a lazily sampled T and a generated msk. We set the DI sampler $\mathcal{S}_6$ to run the PRIV sampler $\mathcal{S}$ and on top of the output keywords, also ask for obfuscations of *messages* that match a first-phase query. By legitimacy of $\mathcal{A}$, these messages must be LR-identical. Using the symmetry of roles for keywords and messages in point functions, this sampler can be shown to be unpredictable whenever the PRIV sampler is. We find paired messages using the image matrix. The obfuscations of messages will allow us to *check* if any of these messages (hidden under the obfuscation) match a first-phase TGen query. For messages that were queried during the first stage, we select the correct T-value. For messages that are at the same time new and paired, we sample a new T-value the first time and answer consistently throughout the game. For all unpaired messages that were never queried to TGen, we use forgetful values (rules of Game$_5$). Second phase TGen queries are answered using T-consistent values relying on the fact that we can use the obfuscations to check matches with paired keywords and the restriction that adversary cannot query a new unpaired message to TGEN. Finally, we output whatever the PRIV adversary $\mathcal{A}$ outputs. More details on how to construct $\mathcal{S}_6, \mathcal{B}_6$ are available in Figure A.3.

$$\Pr[\mathrm{Game}_5(1^\lambda)] - \Pr[\mathrm{Game}_6(1^\lambda)] \leq \Pr[\mathrm{Game}_5(1^\lambda)] - \frac{1}{2} = \frac{1}{2} \cdot \mathbf{Adv}^{\mathrm{di}}_{\mathsf{Obf},\mathcal{S}_6,\mathcal{B}_6}(\lambda) \ .$$

It remains to show that $\mathcal{S}_6$ is unpredictable if $\mathcal{S}$ is. For this, we build a predictor $\mathcal{Q}$ against sampler $\mathcal{S}$, from a predictor $\mathcal{P}$ against sampler $\mathcal{S}_6$ (bottom of Figure A.3). Since $\mathcal{S}_6$ outputs the same vector of circuits as $\mathcal{S}$ plus circuits that are LR-identical, a distinguishing message for the output of $\mathcal{S}_6$ is also a distinguishing message for the output of $\mathcal{S}$. Therefore, we have that

$$\mathbf{Adv}^{\mathrm{pred}}_{\mathcal{S},\mathcal{Q}}(\lambda) = \mathbf{Adv}^{\mathrm{pred}}_{\mathcal{S}_6,\mathcal{P}}(\lambda) \ .$$

To conclude our proof, we put everything together:

$$\begin{aligned}
\mathbf{Adv}^{\text{res-priv}}_{\mathsf{DOX},\mathcal{S},\mathcal{A}}(\lambda) &:= 2 \cdot \Pr[\mathrm{Game}_0(1^\lambda)] - 1 \\
&\leq 2 \cdot \mathbf{Adv}^{\mathsf{prp}}_{\mathsf{PRP},\mathcal{B}_1}(\lambda) + 2 \cdot (t+s+q) \cdot \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{Obf},\mathcal{B}_2}(\lambda) + \\
&\quad 2 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{FE},\mathcal{B}_3}(\lambda) + 2 \cdot \mathbf{Adv}^{\mathrm{di}}_{\mathsf{Obf},\mathcal{S}_4,\mathcal{B}_4}(\lambda) + 2 \cdot (\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{FE},\mathcal{B}_5}(\lambda) + \\
&\quad \frac{s \cdot q}{|\mathsf{WSp}_\lambda|}) + \mathbf{Adv}^{\mathrm{di}}_{\mathsf{Obf},\mathcal{S}_6,\mathcal{B}_6}(\lambda) \ .
\end{aligned}$$

□

```
$\underline{\mathcal{S}_6(1^\lambda, st)}$:
$(st', \mathsf{FirstPhase}) \leftarrow st$
$(\mathbf{w}_0, \mathbf{w}_1, \mathbf{m}_0, \mathbf{m}_1, z') \leftarrow\$ \mathcal{S}(1^\lambda, st)$
for $i \in \{1, \ldots, s\}$
    if $(\mathbf{m}_0[i] \in \mathsf{FirstPhase})$
        // by legitimacy of $\mathcal{A}$, $\mathbf{m}_0[i] \in \mathsf{FirstPhase} \Rightarrow \mathbf{m}_0[i] = \mathbf{m}_1[i]$
        // the following line ensures unpredictability of $\mathcal{S}_6$
        if $(\mathbf{m}_0[i] \neq \mathbf{m}_1[i])$ return $([], [], \perp)$
        $\mathbf{m}^\star \leftarrow \mathbf{m}^\star : \mathbf{m}_0[i]$
    else
        $\mathbf{m}^\star \leftarrow \mathbf{m}^\star : \perp$ // $\mathsf{C}[\perp](.) := 0$ is the zero circuit
$z \leftarrow (z', \mathbf{C}[\mathbf{w}_0](\mathbf{m}_0))$
return $(\mathbf{w}_0 : \mathbf{m}^\star, \mathbf{w}_1 : \mathbf{m}^\star, z)$
```

```
$\underline{\mathcal{B}_6(1^\lambda)}$:
$(\mathsf{msk}, \mathsf{mpk}) \leftarrow\$ \mathsf{FE.Gen}(1^\lambda)$
$b' \leftarrow\$ \mathcal{A}^{\mathrm{LR},\mathrm{TGEN}}(\mathsf{mpk})$
return $b'$
```

```
$\underline{\mathrm{LR}(st)}$:
$\mathsf{FirstPhase} \leftarrow \mathsf{List}$
$(\overline{\mathbf{C}}, z) \leftarrow\$ \mathrm{DI.SAM}((st, \mathsf{FirstPhase}))$
$(z', \mathsf{ImgMatrix}) \leftarrow z$
for $i \in \{1, \ldots, s\}$
    $\mathsf{flag} \leftarrow 0$
    if $(\mathsf{ImgMatrix}[i][] \neq [0, \ldots, 0])$
        $\mathsf{flag} \leftarrow 1$ // $\mathbf{m}_b[i] \in \mathbf{w}_b$
    for $\mathsf{w} \in \mathsf{FirstPhase}$
        if $(\overline{\mathbf{C}}[(t+i)](\mathsf{w}) = 1)$
            $\mathsf{flag} \leftarrow 2$ // $\mathbf{m}_b[i] \in \mathsf{FirstPhase}$
            $\mathsf{m}^\star \leftarrow \mathsf{w}$ // $\mathbf{m}_b[i] = \mathsf{w}$
    if $(\mathsf{flag} = 0)$ // encrypt random message
        $\mathsf{r} \leftarrow\$ \mathsf{WSp}_\lambda$
        $\mathsf{c} \leftarrow\$ \mathsf{FE.Enc}(\mathsf{mpk}, \mathsf{r})$
    if $(\mathsf{flag} = 1)$ // encrypt $T$-consistent
        $(\ldots)$
    if $(\mathsf{flag} = 2)$ // encrypt $T[\mathsf{m}^\star]$
        $\mathsf{c} \leftarrow\$ \mathsf{FE.Enc}(\mathsf{mpk}, T[\mathsf{m}^\star])$
    $\mathbf{c} \leftarrow \mathbf{c} : \mathsf{c}$
$(\ldots)$
return $(\mathbf{tk}, \mathbf{c}, z')$
```

```
$\underline{\mathrm{TGEN}(\mathsf{w})}$:
$\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
if $T[\mathsf{w}] = \perp$ then
    $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
    $\mathsf{List} \leftarrow \mathsf{List} : \mathsf{w}$
$\mathsf{flagRandom} \leftarrow 0$
for $i \in \{1, \ldots, t\}$
    if $(\overline{\mathbf{C}}[i](\mathsf{w}) = 1)$
        // $\mathsf{w} \in \mathbf{w}_b$
        if $(\mathsf{ImgMatrix}[][i] = [0, \ldots, 0])$
            // $\mathsf{w} \notin \mathbf{m}_b$
            if $\mathsf{w} \notin \mathsf{FirstPhase}$
                $\mathsf{flagRandom} \leftarrow 1$
if $(\mathsf{flagRandom} = 1)$
    $\mathsf{r} \leftarrow\$ \mathsf{WSp}_\lambda$
    $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{r}])$
else
    $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
$\mathsf{tk} \leftarrow\$ \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $\mathsf{tk}$
```

```
$\underline{\mathcal{Q}_1(1^\lambda)}$:
$(st, st') \leftarrow\$ \mathcal{P}_1(1^\lambda)$
$(st'', \mathsf{FirstPhase}) \leftarrow st$
return $(st'', (st', \mathsf{FirstPhase}))$
```

```
$\underline{\mathcal{Q}_2^{\mathrm{FUNC}}(1^\lambda, \mathbf{C}[\mathbf{w}_0](\mathbf{m}_0), z', (st', \mathsf{FirstPhase}))}$:
$\mathsf{m} \leftarrow\$ \mathcal{P}_2^{\mathrm{FUNC}'}(1^\lambda, \epsilon, (z', \mathbf{C}[\mathbf{w}_0](\mathbf{m}_0)), st')$
return $\mathsf{m}$

$\underline{\mathrm{FUNC}'(\mathsf{m}, -)}$:
$(\mathbf{C}[\mathbf{w}_0](\mathsf{m}), -) \leftarrow \mathrm{FUNC}(\mathsf{m}, -)$
if $(\mathsf{m} \in \mathsf{FirstPhase})$
    $(-, \mathsf{C}[\mathsf{m}](\mathbf{m}_0)) \leftarrow \mathrm{FUNC}(-, \mathsf{m})$ // if this query is not legit, $\mathbf{Adv}^{\mathrm{pred}}_{\mathcal{S}_6,\mathcal{P}}(\lambda) = 0$
    return $(\mathbf{C}[\mathbf{w}_0](\mathsf{m}) : \mathsf{C}[\mathsf{m}](\mathbf{m}_0))$
else
    return $(\mathbf{C}[\mathbf{w}_0](\mathsf{m}) : [0, \ldots, 0])$
```

Figure A.3: DI adversary $(\mathcal{S}_6, \mathcal{B}_6)$ and predictor $\mathcal{Q}$, as part of proof of Theorem 8.

```
$\mathrm{Game}_0(1^\lambda)$:
$(\mathsf{msk}, \mathsf{mpk}) \leftarrow\$ \mathsf{FE.Gen}(1^\lambda)$
$\mathsf{k} \leftarrow\$ \mathsf{KSp}(1^\lambda)$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\mathrm{LR},\mathrm{TGen}}(\mathsf{mpk})$
return $(b = b')$
```

```
$\mathrm{LR}(st)$:
$(\mathbf{w}_0, \mathbf{w}_1, \mathbf{m}_0, \mathbf{m}_1, z) \leftarrow\$ \mathcal{S}(st)$
$\mathbf{c} \leftarrow\$ \mathsf{FE.Enc}(\mathsf{mpk}, \mathbf{m}_b)$
for all $\mathsf{w} \in \mathbf{w}_b$
  $\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
  $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{E}(\mathsf{k}, \mathsf{w})])$
  $\mathbf{tk} \leftarrow \mathbf{tk} : \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $(\mathbf{tk}, \mathbf{c}, z)$
```

```
$\mathrm{TGen}(\mathsf{w})$:
$\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
$\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{E}(\mathsf{k}, \mathsf{w})])$
$\mathsf{tk} \leftarrow\$ \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $\mathsf{tk}$
```

```
$\mathrm{Game}_1(1^\lambda)$:
$(\mathsf{msk}, \mathsf{mpk}) \leftarrow\$ \mathsf{FE.Gen}(1^\lambda)$
~~$\mathsf{k} \leftarrow\$ \mathsf{KSp}(1^\lambda)$~~
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\mathrm{LR},\mathrm{TGen}}(\mathsf{mpk})$
return $(b = b')$
```

```
$\mathrm{LR}(st)$:
$(\mathbf{w}_0, \mathbf{w}_1, \mathbf{m}_0, \mathbf{m}_1, z) \leftarrow\$ \mathcal{S}(st)$
for all $\mathsf{w} \in \mathbf{w}_b$
  if $T[\mathsf{w}] = \perp$ then
    $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
$\mathbf{c} \leftarrow\$ \mathsf{FE.Enc}(\mathsf{mpk}, \mathbf{m}_b)$
for all $\mathsf{w} \in \mathbf{w}_b$
  $\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
  $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
  $\mathbf{tk} \leftarrow \mathbf{tk} : \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $(\mathbf{tk}, \mathbf{c}, z)$
```

```
$\mathrm{TGen}(\mathsf{w})$:
$\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
if $T[\mathsf{w}] = \perp$ then
  $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
$\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
$\mathsf{tk} \leftarrow\$ \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $\mathsf{tk}$
```

```
$\mathrm{Game}_2(1^\lambda)$:
$(\mathsf{msk}, \mathsf{mpk}) \leftarrow\$ \mathsf{FE.Gen}(1^\lambda)$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\mathrm{LR},\mathrm{TGen}}(\mathsf{mpk})$
if $(\exists\ \mathsf{w}_1, \mathsf{w}_2 \in \mathsf{List}$ s.t. $\mathsf{w}_1 = T[\mathsf{w}_2])$ abort
return $(b = b')$
```

```
$\mathrm{LR}(st)$:
$(\mathbf{w}_0, \mathbf{w}_1, \mathbf{m}_0, \mathbf{m}_1, z) \leftarrow\$ \mathcal{S}(st)$
for all $\mathsf{w} \in \mathbf{w}_b \cup \mathbf{m}_b$
  if $T[\mathsf{w}] = \perp$ then
    $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
    $\mathsf{List} \leftarrow \mathsf{List} : \mathsf{w}$
$\mathbf{c} \leftarrow\$ \mathsf{FE.Enc}(\mathsf{mpk}, \mathbf{m}_b)$
for all $\mathsf{w} \in \mathbf{w}_b$
  $\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
  $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
  $\mathbf{tk} \leftarrow \mathbf{tk} : \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $(\mathbf{tk}, \mathbf{c}, z)$
```

```
$\mathrm{TGen}(\mathsf{w})$:
$\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
if $T[\mathsf{w}] = \perp$ then
  $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
  $\mathsf{List} \leftarrow \mathsf{List} : \mathsf{w}$
$\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
$\mathsf{tk} \leftarrow\$ \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $\mathsf{tk}$
```

```
$\mathrm{Game}_3(1^\lambda)$:
$(\mathsf{msk}, \mathsf{mpk}) \leftarrow\$ \mathsf{FE.Gen}(1^\lambda)$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\mathrm{LR},\mathrm{TGen}}(\mathsf{mpk})$
if $(\exists\ \mathsf{w}_1, \mathsf{w}_2 \in \mathsf{List}$ s.t. $\mathsf{w}_1 = T[\mathsf{w}_2])$ abort
return $(b = b')$
```

```
$\mathrm{LR}(st)$:
$(\mathbf{w}_0, \mathbf{w}_1, \mathbf{m}_0, \mathbf{m}_1, z) \leftarrow\$ \mathcal{S}(st)$
for all $\mathsf{w} \in \mathbf{w}_b \cup \mathbf{m}_b$
  if $T[\mathsf{w}] = \perp$ then
    $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
    $\mathsf{List} \leftarrow \mathsf{List} : \mathsf{w}$
$\mathbf{c} \leftarrow\$ \mathsf{FE.Enc}(\mathsf{mpk}, T[\mathbf{m}_b])$
for all $\mathsf{w} \in \mathbf{w}_b$
  $\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
  $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
  $\mathbf{tk} \leftarrow \mathbf{tk} : \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $(\mathbf{tk}, \mathbf{c}, z)$
```

```
$\mathrm{TGen}(\mathsf{w})$:
$\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
if $T[\mathsf{w}] = \perp$ then
  $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
  $\mathsf{List} \leftarrow \mathsf{List} : \mathsf{w}$
$\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
$\mathsf{tk} \leftarrow\$ \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $\mathsf{tk}$
```

Figure A.4: Sequence of games in proof of Theorem 8 (part 1 of 2).

```
$\underline{\text{Game}_4(1^\lambda)}$:
$(\mathsf{msk}, \mathsf{mpk}) \leftarrow\$ \mathsf{FE.Gen}(1^\lambda)$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\text{LR},\text{TGen}}(\mathsf{mpk})$
if $(\exists\ \mathsf{w}_1, \mathsf{w}_2 \in \mathsf{List}$ s.t. $\mathsf{w}_1 = T[\mathsf{w}_2])$ abort
return $(b = b')$
```

```
$\underline{\text{LR}(st)}$:
$\mathsf{FirstPhase} \leftarrow \mathsf{List}$
$(\mathbf{w}_0, \mathbf{w}_1, \mathbf{m}_0, \mathbf{m}_1, z) \leftarrow\$ \mathcal{S}(st)$
for all $\mathsf{w} \in \mathbf{w}_b \cup \mathbf{m}_b$
  if $T[\mathsf{w}] = \perp$ then
    $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
    $\mathsf{List} \leftarrow \mathsf{List} : \mathsf{w}$
$\mathbf{c} \leftarrow\$ \mathsf{FE.Enc}(\mathsf{mpk}, T[\mathbf{m}_b])$
for all $\mathsf{w} \in \mathbf{w}_b$
  $\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
  if $(\mathsf{w} \notin \mathbf{m}_b \wedge \mathsf{w} \notin \mathsf{FirstPhase})$ then
    $\mathsf{r} \leftarrow\$ \mathsf{WSp}_\lambda$
    $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{r}])$
  else
    $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
  $\mathbf{tk} \leftarrow \mathbf{tk} : \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $(\mathbf{tk}, \mathbf{c}, z)$
```

```
$\underline{\text{TGen}(\mathsf{w})}$:
$\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
if $T[\mathsf{w}] = \perp$ then
  $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
  $\mathsf{List} \leftarrow \mathsf{List} : \mathsf{w}$
if $(\mathsf{w} \in \mathbf{w}_b \wedge \mathsf{w} \notin \mathbf{m}_b \wedge \mathsf{w} \notin \mathsf{FirstPhase})$
  $\mathsf{r} \leftarrow\$ \mathsf{WSp}_\lambda$
  $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{r}])$
else
  $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
$\mathsf{tk} \leftarrow\$ \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $\mathsf{tk}$
```

```
$\underline{\text{Game}_5(1^\lambda)}$:
$(\mathsf{msk}, \mathsf{mpk}) \leftarrow\$ \mathsf{FE.Gen}(1^\lambda)$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\text{LR},\text{TGen}}(\mathsf{mpk})$
if $(\exists\ \mathsf{w}_1, \mathsf{w}_2 \in \mathsf{List}$ s.t. $\mathsf{w}_1 = T[\mathsf{w}_2])$ abort
return $(b = b')$
```

```
$\underline{\text{LR}(st)}$:
$\mathsf{FirstPhase} \leftarrow \mathsf{List}$
$(\mathbf{w}_0, \mathbf{w}_1, \mathbf{m}_0, \mathbf{m}_1, z) \leftarrow\$ \mathcal{S}(st)$
for all $\mathsf{w} \in \mathbf{w}_b \cup \mathbf{m}_b$
  if $T[\mathsf{w}] = \perp$ then
    $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
    $\mathsf{List} \leftarrow \mathsf{List} : \mathsf{w}$
for all $\mathsf{m} \in \mathbf{m}_b$
  if $(\mathsf{m} \notin \mathbf{w}_b \wedge \mathsf{m} \notin \mathsf{FirstPhase})$ then
    $\mathsf{r} \leftarrow\$ \mathsf{WSp}_\lambda$
    $\mathsf{c} \leftarrow\$ \mathsf{FE.Enc}(\mathsf{mpk}, \mathsf{r})$
  else
    $\mathsf{c} \leftarrow\$ \mathsf{FE.Enc}(\mathsf{mpk}, T[\mathsf{m}])$
  $\mathbf{c} \leftarrow \mathbf{c} : \mathsf{c}$
for all $\mathsf{w} \in \mathbf{w}_b$
  $\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
  if $(\mathsf{w} \notin \mathbf{m}_b \wedge \mathsf{w} \notin \mathsf{FirstPhase})$ then
    $\mathsf{r} \leftarrow\$ \mathsf{WSp}_\lambda$
    $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{r}])$
  else
    $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
  $\mathbf{tk} \leftarrow \mathbf{tk} : \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $(\mathbf{tk}, \mathbf{c}, z)$
```

```
$\underline{\text{TGen}(\mathsf{w})}$:
$\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
if $T[\mathsf{w}] = \perp$ then
  $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
  $\mathsf{List} \leftarrow \mathsf{List} : \mathsf{w}$
if $(\mathsf{w} \in \mathbf{w}_b \wedge \mathsf{w} \notin \mathbf{m}_b \wedge \mathsf{w} \notin \mathsf{FirstPhase})$
  $\mathsf{r} \leftarrow\$ \mathsf{WSp}_\lambda$
  $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{r}])$
else
  $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
$\mathsf{tk} \leftarrow\$ \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $\mathsf{tk}$
```

```
$\underline{\text{Game}_6(1^\lambda)}$:
$(\mathsf{msk}, \mathsf{mpk}) \leftarrow\$ \mathsf{FE.Gen}(1^\lambda)$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\text{LR},\text{TGen}}(\mathsf{mpk})$
if $(\exists\ \mathsf{w}_1, \mathsf{w}_2 \in \mathsf{List}$ s.t. $\mathsf{w}_1 = T[\mathsf{w}_2])$ abort
return $(b = b')$
```

```
$\underline{\text{LR}(st)}$:
$\mathsf{FirstPhase} \leftarrow \mathsf{List}$
$(\mathbf{w}_0, \mathbf{w}_1, \mathbf{m}_0, \mathbf{m}_1, z) \leftarrow\$ \mathcal{S}(st)$
for all $\mathsf{w} \in \mathbf{w}_1 \cup \mathbf{m}_1$
  if $T[\mathsf{w}] = \perp$ then
    $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
    $\mathsf{List} \leftarrow \mathsf{List} : \mathsf{w}$
for all $\mathsf{m} \in \mathbf{m}_1$
  if $(\mathsf{m} \notin \mathbf{w}_1 \wedge \mathsf{m} \notin \mathsf{FirstPhase})$ then
    $\mathsf{r} \leftarrow\$ \mathsf{WSp}_\lambda$
    $\mathsf{c} \leftarrow\$ \mathsf{FE.Enc}(\mathsf{mpk}, \mathsf{r})$
  else
    $\mathsf{c} \leftarrow\$ \mathsf{FE.Enc}(\mathsf{mpk}, T[\mathsf{m}])$
  $\mathbf{c} \leftarrow \mathbf{c} : \mathsf{c}$
for all $\mathsf{w} \in \mathbf{w}_1$
  $\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
  if $(\mathsf{w} \notin \mathbf{m}_1 \wedge \mathsf{w} \notin \mathsf{FirstPhase})$ then
    $\mathsf{r} \leftarrow\$ \mathsf{WSp}_\lambda$
    $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{r}])$
  else
    $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
  $\mathbf{tk} \leftarrow \mathbf{tk} : \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $(\mathbf{tk}, \mathbf{c}, z)$
```

```
$\underline{\text{TGen}(\mathsf{w})}$:
$\overline{\mathsf{C}}_\mathsf{L} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{w}])$
if $T[\mathsf{w}] = \perp$ then
  $T[\mathsf{w}] \leftarrow\$ \mathsf{WSp}_\lambda \setminus T$
  $\mathsf{List} \leftarrow \mathsf{List} : \mathsf{w}$
if $(\mathsf{w} \in \mathbf{w}_1 \wedge \mathsf{w} \notin \mathbf{m}_1 \wedge \mathsf{w} \notin \mathsf{FirstPhase})$
  $\mathsf{r} \leftarrow\$ \mathsf{WSp}_\lambda$
  $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[\mathsf{r}])$
else
  $\overline{\mathsf{C}}_\mathsf{R} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathsf{C}[T[\mathsf{w}]])$
$\mathsf{tk} \leftarrow\$ \mathsf{FE.TGen}(\mathsf{msk}, (\overline{\mathsf{C}}_\mathsf{L} \vee \overline{\mathsf{C}}_\mathsf{R}))$
return $\mathsf{tk}$
```

Figure A.5: Sequence of games in proof of Theorem 8 (part 2 of 2).