Towards a Unified Theory of Cryptographic Agents

Shashank Agrawal [*] Shweta Agrawal [†] Manoj Prabhakaran [‡]

Abstract

In recent years there has been a fantastic boom of increasingly sophisticated "cryptographic objects" — identity-based encryption, fully-homomorphic encryption, functional encryption, and most recently, various forms of obfuscation. These objects often come in various flavors of security, and as these constructions have grown in number, complexity and inter-connectedness, the relationships between them have become increasingly confusing.

We provide a new framework of *cryptographic agents* that unifies various cryptographic objects and security definitions, similar to how the Universal Composition framework unifies various multi-party computation tasks like commitment, coin-tossing and zero-knowledge proofs.

Our contributions can be summarized as follows.

- Our main contribution is a new model of cryptographic computation, that unifies and extends cryptographic primitives such as Obfuscation, Functional Encryption, Fully Homomorphic Encryption, Witness encryption, Property Preserving Encryption and the like, all of which can be cleanly modeled as "schemata" in our framework. We provide a new *indistinguishability preserving* (IND-PRE) definition of security that interpolates indistinguishability and simulation style definitions, implying the former while (often) sidestepping the impossibilities of the latter.
- We present a notion of *reduction* from one schema to another and a powerful *composition theorem* with respect to IND-PRE security. This provides a modular means to build and analyze secure schemes for complicated schemata based on those for simpler schemata. Further, this provides a way to abstract out and study the relative complexity of different schemata. We show that obfuscation is a "complete" schema under this notion, under standard cryptographic assumptions.
- IND-PRE-security can be parameterized by the choice of the "test" family. For obfuscation, the strongest security definition (by considering all PPT tests) is unrealizable in general. But we identify a family of tests, called Δ, such that all known impossibility results, for obfuscation as well as functional encryption, are by-passed. On the other hand, for each of the example primitives we consider in this paper – obfuscation, functional encryption, fully-homomorphic encryption and property-preserving encryption – Δ-IND-PRE-security for the corresponding schema implies the standard achievable security definitions in the literature.
- We provide a stricter notion of reduction that composes with respect to Δ-IND-PRE-security.
- Based on Δ-IND-PRE-security we obtain a new definition for security of obfuscation, called *adaptive differing-inputs* obfuscation. We illustrate its power by using it for new constructions of functional encryption schemes, with and without function-hiding.
- Last but not the least, our framework can be used to model abstractions like the generic group model and the random oracle model, letting one translate a general class of constructions in these heuristic models to constructions based on *standard model assumptions*. We illustrate this by adapting a functional encryption scheme (for inner product predicate) that was shown secure in the generic group model to be secure based on a new standard model assumption we propose, called the *generic bilinear group agents* assumption.

[*]University of Illinois, Urbana-Champaign. Email: `sagrawl2@illinois.edu`.
[†]Indian Institute of Technology, Delhi. Email: `shweta.a@gmail.com`.
[‡]University of Illinois, Urbana-Champaign. Email: `mmp@illinois.edu`.

Contents

1 Introduction

Over the last decade or so, thanks to remarkable breakthroughs in cryptographic techniques, a wave of "cryptographic objects" — identity-based encryption, fully-homomorphic encryption, functional encryption, and most recently, various forms of obfuscation — have opened up exciting new possibilities for cryptographers. Initial foundational results on this front consisted of strong impossibility results. Breakthrough constructions, as they emerged, often used specialized security definitions which avoided such impossibility results. However, as these objects and their constructions have become numerous and complex, sometimes building on each other, the connections among these disparate cryptographic objects, and among their disparate security definitions, remain not well understood.

The goal of this work is to provide a clean and unifying framework for various cryptographic objects and security definitions, equipped with powerful *reductions* and *composition theorems*. By choosing the nature of the "test" in our framework (e.g., randomized or deterministic), we obtain various existing and new flavors of security definitions for functional encryption and obfuscation. Some of the resulting definitions are provably impossible to achieve, but stronger versions that avoid known impossibility results also emerge (e.g., a new notion of "adaptive differing-inputs obfuscation" that leads to significant simplifications in constructions using "differing-inputs obfuscation").

Another important contribution of our framework is that it can be used to model abstractions like the generic group model, letting one translate a general class of constructions in these heuristic models to constructions based on *standard model assumptions*.

Cryptographic Agents. Our unifying framework, called the *Cryptographic Agents framework* models one or more (possibly randomized, stateful) objects that interact with each other, so that a user with access to their codes can only learn what it can learn from the output of these objects. As a running example, functional encryption schemes could be considered as consisting of "message agents" and "key agents."

To formalize the security requirement, we use a real-ideal paradigm, but at the same time rely on an indistinguishability notion (rather than a simulation-based security notion). We informally describe the framework below. Figure 1 illustrates the real and ideal executions.

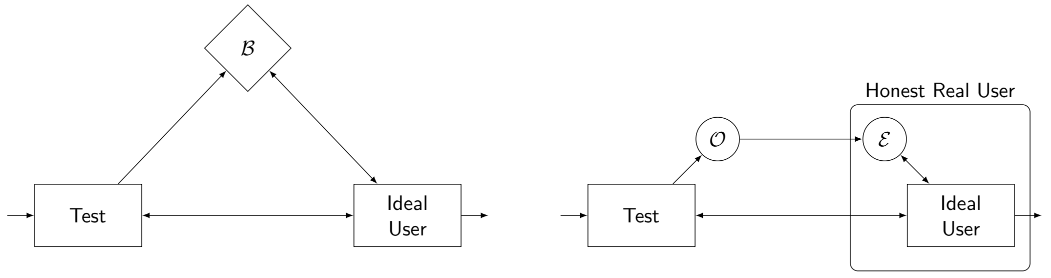

Figure 1 The ideal world (on the left) and the real world with an honest user.

- **Ideal Execution.** The ideal world consists of two (adversarially designed) entities — a User and a Test — who can freely interact with each other. User is given access, via handles, to a collection of "agents" (interactive Turing Machines), maintained by $\mathcal{B}$ (a "blackbox"). User and Test are both allowed to add agents to the collection of agents with $\mathcal{B}$, but the class of agents that they can add are restricted by an *ideal agents schema*.[1] For example, in a schema capturing public-key functional encryption, the user can add only "message agents" each of which simply copies an inbuilt message into its communication tape everytime it is invoked; but Test can also add "key agents," each of

[1]Here, a *schema* is analogous to a *functionality* in UC security. Thus different primitives like functional encryption and fully-homomorphic encryption are specified by different schemata.

which reads a message from its incoming communication tape, applies an inbuilt function to it, and copies the result to its output tape. In a general schema, the User can feed inputs to these agents, and also allow a set of them to interact with each other, in a "session". At the end of this interaction, the user obtains all the outputs from the session, and also additional handles to the agents with updated states.[2]

- **Real Execution.** The real execution also consists of two entities, the (real-world) user (or an adversary Adv) and Test. The latter is in fact the same as in the ideal world. But in the real world, when Test requests adding an agent to the collection of agents, the user is handed a cryptographically generated object – a "cryptographic agent" – instead of a handle to this agent. The *correctness requirement* is that an honest user should be able to perform all the operations any User can in the ideal world (i.e., add new agents to the collection, and execute a session of agents, and thereby update their states) using an "execution" operation applied to the cryptographic agents. In Figure 1, $\mathcal{O}$ indicates the algorithm for encoding, and $\mathcal{E}$ indicates a procedure that applies an algorithm for session executions, as requested by the User. (However, an adversarial user Adv in the real world may analyze the cryptographic agents in anyway it wants.)
- **Security Definition.** We define IND-PRE (for indistinguishability preserving) security, which requires that *if* a Test is such that a certain piece of information about it (modeled as an input bit) remains hidden from every user in the ideal world, *then* that information should stay hidden from every user that interacts with Test in the real world as well. Note that we do not require that the view in the real world can be simulated in the ideal world.

  In the real world we require all entities to be computationally bounded. But in the *ideal* world, we may consider users that are computationally bounded or unbounded (possibly with a limit on the number of sessions it can invoke). We may also restrict ourselves to deterministic Tests. These choices allows us to model different levels of security, which in the case of specific schema, translate to various natural notions of security.

1.1 Our Contributions

Our main contribution is a new model of cryptographic computation, that unifies and extends emerging concepts in the field. Obfuscation, Functional Encryption, Fully Homomorphic Encryption, Property Preserving Encryption, etc. can all be easily and cleanly modeled by specific schemata in our model.

One can consider our framework analogous to the now-standard approach in secure multi-party computation (MPC) (e.g., following [57, 36]) that uses *a common paradigm to abstract the security guarantees in a variety of different tasks* like commitments, zero-knowledge proofs, coin-flipping, oblivious-transfer etc. Such a common framework allows one not only to see the connections across primitives, but also to identify better definitions that increase the usability and security of the primitives. While we anticipate several refinements and extensions to the framework presented here, we consider that, thanks to its simplicity, the current model already provides important insight about the "right" security notions for the latest set of primitives in modern cryptography, and opens up a wealth of new questions and connections for further investigation.

The list of technical results in this paper could be viewed in two parts: contributions to the foundational aspects of cryptographic objects, and contributions to specific objects of interest (mainly, obfuscation and functional encryption). Some of our specific contributions to the foundational aspects of this area are as follows.

- We first define a general framework of cryptographic agents that can be instantiated for different

[2]For our example of functional encryption, neither inputs nor states are relevant, as the message and key agents have all the relevant information built in. However, obfuscation is best modeled by non-interactive agents that take an input, and modeling fully homomorphic encryption requires agents that maintain state.

primitives using different *schemata*. The resulting security definition, called Γ-$\mathsf{IND\text{-}PRE}$-security is paramterized by a test family Γ. For natural choices of Γ, these definitions tend to be not only stronger than standard definitions, but also easier to work with in larger constructions (see next).

- We present a notion of reduction from one schema to another,[3] and a composition theorem. This provides a way to study, in abstract, relative complexity of different schemata – in particular, showing that obfuscation is a "complete" schema under this notion. But further, it provides a modular means to build and analyze secure schemes for complicated schema based on those for simpler schemata.
- The above notion of reduction composes for Γ_{ppt}-$\mathsf{IND\text{-}PRE}$-security where Γ_{ppt} is the class of all probabilistic polynomial time (PPT) tests. Unfortunately, obfuscation (and hence, any other complete schema) can be shown to be unrealizable under this definition.
- We identify a simple test family Δ (defined later) such that for each of the example primitives we consider in this paper – obfuscation, functional encryption, fully-homomorphic encryption and property-preserving encryption – Δ-$\mathsf{IND\text{-}PRE}$-security for the corresponding schema implies the standard security definitions in the literature (except the simulation-based definitions which are known to be impossible to realize in general).

  We also identify a few natural, restricted test families which yield definitions that are *equivalent* to various existing security definitions in the literature.
- Then we present a more structured notion of reduction, called Δ-reduction, that composes with respect to Δ-$\mathsf{IND\text{-}PRE}$-security as well.

These basic results have several important implications to specific problems of interest.

• **Obfuscation.** We study in detail, the various notions of obfuscation in the literature, and relate them to Γ-$\mathsf{IND\text{-}PRE}$-security for various test families Γ. Our strongest definition of this form, which considers the family of all PPT tests, turns out to be impossible. Our definition is conceptually "weaker" than the virtual black-box simulation definition (in that it does not require a simulator), but the impossibility result of Barak et al. [9] continues to apply to this definition. To circumvent the impossibility, we identify three test families, Δ, Δ^* and Δ_{det}, such that Δ_{det}-$\mathsf{IND\text{-}PRE}$-security is *equivalent*[4] to indistinguishability obfuscation, Δ^*-$\mathsf{IND\text{-}PRE}$-security is equivalent to differing inputs obfuscation, and Δ-$\mathsf{IND\text{-}PRE}$-security implies both the above. We state a new definition for the security of obfuscation – *adaptive differing-inputs obfuscation* – which is equivalent Δ-$\mathsf{IND\text{-}PRE}$-security. Informally, it is the same as differing inputs obfuscation, but an adversary is allowed to *interact* with the "sampler" (which samples two circuits one of which will be obfuscated and presented to the adversary as a challenge), even after it receives the obfuscation.

• **Functional Encryption.** Various flavors of functional encryption (FE) have been considered in the literature – public and private key, with or without function hiding – with various formalizations of the security requirement – indistinguishability based or simulation based security, further classified into adaptive and non-adaptive security, possibly with a priori bounds on the number of keys issued. Our framework provides a unified method to capture all these variants, using slightly different schemata and test families. For concreteness, we focus on adaptive secure, indistinguishability-based, public-key FE (with and without function-hiding).

– We present a schema $\mathbf{\Sigma}_{\text{FE}}$ which captures FE without function-hiding. We consider a hierarchy of security notions Δ_{det}-$\mathsf{IND\text{-}PRE} \leq \Delta$-$\mathsf{IND\text{-}PRE} \leq \mathsf{IND\text{-}PRE} \leq \mathsf{SIM}$ and show that Δ_{det}-$\mathsf{IND\text{-}PRE}$ FE is equivalent to the standard indistinguishability or IND based notion of security (in contrast SIM

[3]Our reduction uses a simulation-based security requirement. Thus, among other things, it also provides a means for capturing simulation-based security definition: we say that a scheme Π is a Γ-SIM-secure scheme for a schema $\mathbf{\Sigma}$ if Π reduces $\mathbf{\Sigma}$ to the null-schema.

[4]Equivalences are upto easily bridged syntactic differences between an obfuscation scheme and a cryptographic agent.

security is known to be impossible for general function families [21, 13, 5]). For function-hiding FE too, IND-PRE-security is impossible for general function families, since such FE subsumes obfuscation.

– Function-hiding (public-key) FE had proved difficult to define satisfactorily [19, 20, 1]. The IND-PRE framework provides a way to obtain a natural and general definition of this primitive. We present a schema $\mathbf{\Sigma}_{\text{FH-FE}}$ to capture the security guarantees of function-hiding FE.
– We present new constructions for Δ-IND-PRE secure FE (both with and without function hiding) for all polynomial-time computable functions. We also present an IND-PRE secure FE for the inner product functionality.

We remark that two of our three constructions are in the form of reductions (a Δ-reduction to the obfuscation schema, and a (standard) reduction to a "bilinear generic group" schema as described below). Also, the first two constructions crucially rely on Δ-IND-PRE-security of obfuscation (i.e., adaptive differing-inputs obfuscation), thereby considerably simplifying the constructions and the analysis compared to those in recent work [7, 25] which uses (non-adaptive) differing-inputs obfuscation.

• **Using the Generic Group in the Standard Model.** One can model random oracles and the generic group model as schemata. An assumption that such a schema has an IND-PRE-secure scheme is a standard model assumption, and to the best of our knowledge, not ruled out by the techniques in the literature. This is because, IND-PRE-security captures only certain indistinguishability guarantees of the generic group model, albeit in a broad manner (by considering arbitrary tests). Indeed, for random oracles, such an assumption is implied by (for instance) virtual blackbox secure obfuscation of point-functions, a primitive that has plausible candidates in the literature.

The generic group schema (as well as its bilinear version) is a highly versatile resource used in several constructions, including that of cryptographic objects that can be modeled as schemata. Such constructions can be considered as *reductions* to the generic group schema. Combined with our composition theorem, this creates a recipe for standard model constructions under a strong, but simple to state, computational assumption.

We give such an example for obtaining a standard model *function-hiding* public-key FE scheme for inner-product predicates (for which a satisfactory general security definition has also been lacking).

• **Other Primitives.** Our model is extremely flexible, and can easily capture most cryptographic objects for which an indistinguishability security notion is required. This includes witness encryption, functional witness encryption, fully homomorphic encryption (FHE), property-preserving encryption (PPE) etc. We discuss a couple of them – FHE and PPE – to illustrate this. We can model FHE using (stateful) cryptographic agents. The resulting security definition, even with the test family Δ_{det}, implies the standard definition in the literature, with the additional requirement that a ciphertext does not reveal how it was formed, even given the decryption key. For PPE, we show that an Δ_{det}-IND-PRE secure scheme for the PPE schema is in fact equivalent to a scheme that satisfies the standard definition of security for PPE.

We consider this work the first step in formulating a broader theory of active or functional cryptographic objects. Our current results bring out only some of the numerous potential advantages of the proposed framework. On the other hand, there are limitations to its scope. For example, the model in this work does not currently consider security issues arising from maliciously crafted cryptographic agents. In particular, it does not model signature schemes. We leave such an extension for future work.

Related Work. Recently, there has been a tremendous amount of work on objects we model, including FE and obfuscation. We discuss some of it in Appendix A.

2 Preliminaries

To formalize the model of cryptographic agents, we shall use the standard notion of probabilistic interactive Turing Machines (ITM) with some modifications (see below). To avoid cumbersome formalism, we keep the description somewhat informal, but it is straightforward to fully formalize our model. We shall also not attempt to define the model in its most generality, for the sake of clarity.

In our case an ITM has separate tapes for input, output, incoming communication, outgoing communication, randomness and work-space.

Definition 1 (Agents and Family of Agents)**.** *An agent is an interactive Turing Machine, with the following modifications:*

- *There is a special read-only parameter tape, which always consists of a security parameter κ, and possibly other parameters.*
- *There is an a priori restriction on the size of all the tapes other than the randomness tape (including input, communication and work tapes), as a function of the security parameter.*
- *There is a special* blocking state *such that if the machine enters such a state, it remains there if the input tape is empty. Similarly, there are blocking states which let the machine block if any combination of the communication tape and the input tape is empty.*

An agent family *is a maximal set of agents with the same program (i.e., state space and transition functions), but possibly different contents in their parameter tapes. We also allow an agent family to be the empty set $\emptyset$.*

We can allow *non-uniform agents* by allowing an additional advice tape. Our framework and results work in the uniform and non-uniform model equally well.

Note that an agent who enters a blocking state can move out of it if its configuration is changed by adding a message to its input tape and/or communication tape. However, if the agent enters a halting state, it will not move out of that state. An agent who never enters a blocking state is called a *non-reactive agent.* An agent who never reads or writes from a communication tape is called a *non-interactive agent.*

Definition 2 (Session)**.** *A session maps a finite ordered set of agents, their configurations and inputs to outputs and (updated) configurations of the same agents, as follows: the agents are initialized with the given inputs on their input tapes, and then executed together until they are deadlocked.*[5] *The session returns the resulting collection of outputs and configurations of the agents (if the session terminates).*

We shall be restricting ourselves to collections of agents such that sessions involving them are guaranteed to terminate. Note that we have defined a session to have only an initial set of inputs, so that the outcome of a session is well-defined (without the need to specify how further inputs would be chosen).

Next we define an important notion in our framework, namely that of an *ideal agent schema*, or simply, a schema. A schema plays the same role as a functionality does in the Universal Composition framework for secure multi-party computation. That is, it specifies what is legitimate for a user to do in a system. A schema defines the families of agents that the user and a "test" (or authority) are allowed to create.

[5]More precisely, the first agent is executed till it enters a blocking or halting state, and then the second and so forth, in a round-robin fashion, until all the agents have been in blocking or halting states for a full round. After each execution of an agent, the contents of its outgoing communication tape are interpreted as an ordered sequence of messages to each of the other agents in the session (some or all of them possibly being empty messages), and copied over to the respective agents' incoming communication tapes.

Definition 3 (Ideal Agent Schema)**.** *A (well-behaved)* ideal agent schema $\mathbf{\Sigma} = (\mathcal{P}_{\mathsf{auth}}, \mathcal{P}_{\mathsf{user}})$ *(or simply schema) is a pair of agent families, such that there is a polynomial* poly *such that for any session of agents belonging to* $\mathcal{P}_{\mathsf{auth}} \cup \mathcal{P}_{\mathsf{user}}$ *(with any inputs and any configurations, with the same security parameter* κ*), the session terminates within* $\mathrm{poly}(\kappa, t)$ *steps, where* t *is the number of agents in the session.*

We use the short-hand $X \approx Y$ to denote two binary random variables that are distributed almost identically. More precisely, if X and Y are a family of random variables (one for each value of κ), we write $X \approx Y$ if there is a negligible function negl such that $|\Pr[X = 1] - \Pr[Y = 1]| \leq \mathrm{negl}(\kappa)$.

For two systems M and M', we say $M \approxeq M'$ if the two systems are indistinguishable to an interactive PPT distinguisher.

3 Defining Cryptographic Agents

In this section we define what it means for a cryptographic agent scheme to securely implement a given ideal agent schema. At a very high-level, the security notion is of *indistinguishability preservation*: if two executions using an ideal schema are indistinguishable, we require them to remain indistinguishable when implemented using a cryptographic agent scheme. While it consists of several standard elements of security definitions prevalent in modern cryptography, indistinguishability preservation as defined here is novel, and potentially of broader interest.

Ideal World The ideal system for a schema $\mathbf{\Sigma}$ consists of two parties Test and User and a fixed third party $\mathcal{B}[\mathbf{\Sigma}]$ (for "black-box"). All three parties are probabilistic polynomial time (PPT) ITMs, and have a security parameter κ built-in. We shall explicitly refer to their random-tapes as r, s and t. Test receives a "secret bit" b as input and User produces an output bit b'.

The interaction between User, Test and $\mathcal{B}[\mathbf{\Sigma}]$ can be summarized as follows:

- **Uploading agents.** Let $\mathbf{\Sigma} = (\mathcal{P}_{\mathsf{auth}}, \mathcal{P}_{\mathsf{user}})$ where we associate $\mathcal{P}_{\mathsf{test}} := \mathcal{P}_{\mathsf{auth}} \cup \mathcal{P}_{\mathsf{user}}$ with Test and $\mathcal{P}_{\mathsf{user}}$ with User. Test and User can, at any point, choose an agent from its agent family and send it to $\mathcal{B}[\mathbf{\Sigma}]$. More precisely, User can send a string to $\mathcal{B}[\mathbf{\Sigma}]$, and $\mathcal{B}[\mathbf{\Sigma}]$ will instantiate an agent $\mathcal{P}_{\mathsf{user}}$, with the given string (along with its own security parameter) as the contents of the parameter tape, and all other tapes being empty. Similarly, Test can send a string and a bit indicating whether it is a parameter for $\mathcal{P}_{\mathsf{auth}}$ or $\mathcal{P}_{\mathsf{user}}$, and it is used to instantiate an agent $\mathcal{P}_{\mathsf{auth}}$ or $\mathcal{P}_{\mathsf{user}}$, accordingly [6]. Whenever an agent is instantiated, $\mathcal{B}[\mathbf{\Sigma}]$ sends a unique handle (a serial number) for that agent to User; the handle also indicates whether the agent belongs to $\mathcal{P}_{\mathsf{auth}}$ or $\mathcal{P}_{\mathsf{user}}$.
- **Request for Session Execution.** At any point in time, User may request an execution of a session, by sending an ordered tuple of handles $(h_1, \ldots, h_t)$ (from among all the handles obtained thus far from $\mathcal{B}[\mathbf{\Sigma}]$) to specify the configurations of the agents in the session, along with their inputs. $\mathcal{B}[\mathbf{\Sigma}]$ reports back the outputs from the session, and also gives new handles corresponding to the configurations of the agents when the session terminated.[7] If an agent halts in a session, no new handle is given for that agent.

Observe that only User receives any output from $\mathcal{B}[\mathbf{\Sigma}]$; the communication between Test and $\mathcal{B}[\mathbf{\Sigma}]$ is one-way. (See Figure 1.)

We define the random variable $\text{IDEAL}\langle \mathsf{Test}(b) \mid \mathbf{\Sigma} \mid \mathsf{User}\rangle$ to be the output of User in an execution of the above system, when Test gets b as input. We write $\text{IDEAL}\langle \mathsf{Test} \mid \mathbf{\Sigma} \mid \mathsf{User}\rangle$ in the case when the

[6] In fact, for convenience, we allow Test and User to specify multiple agents in a single message to $\mathcal{B}[\mathbf{\Sigma}]$.

[7] Note that if the same handle appears more than once in the tuple $(h_1, \ldots, h_t)$, it is interpreted as multiple agents with the same configuration (but possibly different inputs). Also note that after a session, the old handles for the agents are not invalidated; so a User can access a configuration of an agent any number of times, by using the same handle.

input to Test is a uniformly random bit. We also define $\textsc{Time}\langle \mathsf{Test} \mid \mathbf{\Sigma} \mid \mathsf{User} \rangle$ as the maximum number of steps taken by Test (with a random input), $\mathcal{B}[\mathbf{\Sigma}]$ and User in total.

Definition 4. *We say that* Test *is* hiding w.r.t. $\mathbf{\Sigma}$ *if* $\forall$ *PPT party* User*,*

$$\textsc{ideal}\langle \mathsf{Test}(0) \mid \mathbf{\Sigma} \mid \mathsf{User} \rangle \approx \textsc{ideal}\langle \mathsf{Test}(1) \mid \mathbf{\Sigma} \mid \mathsf{User} \rangle.$$

Real World. A *cryptographic scheme* (or simply scheme) consists of a pair of (possibly stateful and randomized) programs $(\mathcal{O}, \mathcal{E})$, where $\mathcal{O}$ is an encoding procedure for agents in $\mathcal{P}_{\mathsf{test}}$ and $\mathcal{E}$ is an execution procedure. The real world execution for a scheme $(\mathcal{O}, \mathcal{E})$ consists of Test, a user that we shall generally denote as Adv and the encoder $\mathcal{O}$. ($\mathcal{E}$ features as part of an honest user in the real world execution: see Figure 1.) Test remains the same as in the ideal world, except that instead of sending an agent to $\mathcal{B}[\mathbf{\Sigma}]$, it sends it to the encoder $\mathcal{O}$. In turn, $\mathcal{O}$ encodes this agent and sends the resulting cryptographic agent to Adv.

We define the random variable $\textsc{real}\langle \mathsf{Test}(b) \mid \mathcal{O} \mid \mathsf{Adv} \rangle$ to be the output of Adv in an execution of the above system, when Test gets b as input; as before, we omit b from the notation to indicate a random bit. Also, as before, $\textsc{Time}\langle \mathsf{Test} \mid \mathcal{O} \mid \mathsf{User} \rangle$ is the maximum number of steps taken by Test (with a random input), $\mathcal{O}$ and User in total.

Definition 5. *We say that* Test *is* hiding w.r.t. $\mathcal{O}$ *if* $\forall$ *PPT party* Adv*,*

$$\textsc{real}\langle \mathsf{Test}(0) \mid \mathcal{O} \mid \mathsf{Adv} \rangle \approx \textsc{real}\langle \mathsf{Test}(1) \mid \mathcal{O} \mid \mathsf{Adv} \rangle.$$

Note that $\textsc{real}\langle \mathsf{Test} \mid \mathcal{O} \mid \mathsf{Adv} \rangle = \textsc{real}\langle \mathsf{Test} \circ \mathcal{O} \mid \emptyset \mid \mathsf{Adv} \rangle$ where $\emptyset$ stands for the null implementation. Thus, instead of saying Test is hiding w.r.t. $\mathcal{O}$, we shall sometimes say $\mathsf{Test} \circ \mathcal{O}$ is hiding (w.r.t. $\emptyset$). Also, when $\mathcal{O}$ is understood, we may simply say that Test is real-hiding.

Syntactic Requirements on $(\mathcal{O}, \mathcal{E})$. $(\mathcal{O}, \mathcal{E})$ may or may not use a "setup" phase. In the latter case we call it a *setup-free cryptographic agent scheme*, and $\mathcal{O}$ is required to be a memory-less program that takes an agent $P \in \mathcal{P}_{\mathsf{test}}$ as input and outputs a cryptographic agent that is sent to Adv. If the scheme has a setup phase, $\mathcal{O}$ consists of a triplet of memory-less programs $(\mathcal{O}_{\mathsf{setup}}, \mathcal{O}_{\mathsf{auth}}, \mathcal{O}_{\mathsf{user}})$: in the real world execution, first $\mathcal{O}_{\mathsf{setup}}$ is run to generate a secret-public key pair $(\mathsf{MSK}, \mathsf{MPK})$;[8] MPK is sent to Adv. Subsequently, when $\mathcal{O}$ receives an agent $P \in \mathcal{P}_{\mathsf{auth}}$ it will invoke $\mathcal{O}_{\mathsf{auth}}(P, \mathsf{MSK})$, and when it receives an agent $P \in \mathcal{P}_{\mathsf{user}}$, it will invoke $\mathcal{O}_{\mathsf{user}}(P, \mathsf{MPK})$, to obtain a cryptographic agent that is then sent to Adv.

$\mathcal{E}$ is required to be memoryless as well, except that when it gives a handle to a User, it can record a string against that handle, and later when User requests a session execution, $\mathcal{E}$ can access the string recorded for each handle in the session. There is a *compactness requirement* that the size of this string is *a priori* bounded (note that the state space of the ideal agents are also *a priori* bounded). If there is a setup phase, $\mathcal{E}$ can also access MPK each time it is invoked.

IND-PRE Security. Now we are ready to present the security definition of a cryptographic agent scheme $(\mathcal{O}, \mathcal{E})$ implementing a schema $\mathbf{\Sigma}$. Below, the *honest real-world user*, corresponding to an ideal-world user User, is defined as the composite program $\mathcal{E} \circ \mathsf{User}$ as shown in Figure 1.

Definition 6. *A cryptographic agent scheme* $\Pi = (\mathcal{O}, \mathcal{E})$ *is said to be a* Γ-IND-PRE-secure scheme *for a schema* $\mathbf{\Sigma}$ *if the following conditions hold.*

- *Correctness.* $\forall$ *PPT* User, $\forall \mathsf{Test} \in \Gamma$, $\textsc{ideal}\langle \mathsf{Test} \mid \mathbf{\Sigma} \mid \mathsf{User} \rangle \approx \textsc{real}\langle \mathsf{Test} \mid \mathcal{O} \mid \mathcal{E} \circ \mathsf{User} \rangle$. *If equality holds,* $(\mathcal{O}, \mathcal{E})$ *is said to have perfect correctness.*
- *Efficiency. There exists a polynomial* poly *such that,* $\forall$ *PPT* User, $\forall \mathsf{Test} \in \Gamma$,

$$\textsc{Time}\langle \mathsf{Test} \mid \mathcal{O} \mid \mathcal{E} \circ \mathsf{User} \rangle \le \mathrm{poly}(\textsc{Time}\langle \mathsf{Test} \mid \mathbf{\Sigma} \mid \mathsf{User} \rangle, \kappa).$$

[8]For "master" secret and public-keys, following the terminology in some of our examples.

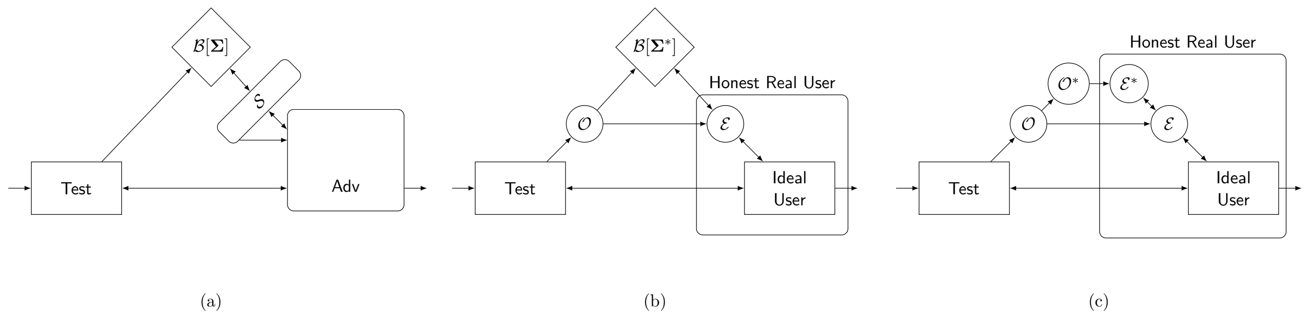

Figure 2 $(\mathcal{O}, \mathcal{E})$ in (b) is a reduction from schema $\mathbf{\Sigma}$ to $\mathbf{\Sigma}^*$. The security requirement is that no adversary Adv in the system (a) can distinguish that execution from an execution of the system in (b) (with Adv taking the place of honest real user). The correctness requirement is that the ideal User in (b) behaves the same as the ideal User interacting directly with $\mathcal{B}[\mathbf{\Sigma}]$ (as in Figure 1(a)). (c) shows the composition of the hybrid scheme $(\mathcal{O}, \mathcal{E})^{\mathbf{\Sigma}^*}$ with a scheme $(\mathcal{O}^*, \mathcal{E}^*)$ that IND-PRE-securely implements $\mathbf{\Sigma}^*$.

- *Indistinguishability Preservation.* $\forall \mathsf{Test} \in \Gamma$,

$$\mathsf{Test} \textit{ is hiding w.r.t. } \mathbf{\Sigma} \Rightarrow \mathsf{Test} \textit{ is hiding w.r.t. } \mathcal{O}.$$

When Γ *is the family of all PPT tests – denoted by* Γ_{ppt}*, we simply say that* Π *is an* IND-PRE*-secure scheme for* $\mathbf{\Sigma}$*.*

When the schema is understood, we shall refer to the property of being hiding w.r.t. a schema as simply being ideal-hiding.

4 Reductions and Compositions

A fundamental question regarding (secure) computational models is that of reduction: which tasks can be reduced to which others. In the context of cryptographic agents, we ask which schemata can be reduced to which other schemata. We shall use a strong *simulation-based* notion of reduction. While a simulation-based security notion for general cryptographic agents or even just obfuscations (i.e., virtual black-box obfuscation) is too strong to exist, it is indeed possible to meet a simulation-based notion for reductions between schemata. This is *analogous to the situation in Universally Composable security*, where sweeping impossibility results exist for UC secure realizations in the plain model, but there is a rich structure of UC secure reductions among functionalities.

A *hybrid scheme* $(\mathcal{O}, \mathcal{E})^{\mathbf{\Sigma}^*}$ is a cryptographic agent scheme in which $\mathcal{O}$ and $\mathcal{E}$ have access to $\mathcal{B}[\mathbf{\Sigma}^*]$, as shown in Figure 2 (in the middle), where $\mathbf{\Sigma}^* = (\mathcal{P}^*_{\mathsf{auth}}, \mathcal{P}^*_{\mathsf{user}})$. If $\mathcal{O}$ has a setup phase, we require that $\mathcal{O}_{\mathsf{user}}$ uploads agents only in $\mathcal{P}^*_{\mathsf{user}}$ (but $\mathcal{O}_{\mathsf{auth}}$ can upload any agent in $\mathcal{P}^*_{\mathsf{auth}} \cup \mathcal{P}^*_{\mathsf{user}}$). In general, the honest user would be replaced by an adversarial user Adv. Note that the output bit of Adv in such a system is given by the random variable $\text{IDEAL}\langle \mathsf{Test} \circ \mathcal{O} \mid \mathbf{\Sigma}^* \mid \mathsf{Adv} \rangle$, where $\mathsf{Test} \circ \mathcal{O}$ denotes the combination of Test and $\mathcal{O}$ as in Figure 2.

Definition 7 (Reduction)**.** *We say that a (hybrid) cryptographic agent scheme* $\Pi = (\mathcal{O}, \mathcal{E})$ reduces $\mathbf{\Sigma}$ *to* $\mathbf{\Sigma}^*$ *with respect to* Γ*, if there exists a simulator* $\mathcal{S}$ *such that* $\forall$ PPT User*,*

1. *Correctness:* $\forall \mathsf{Test} \in \Gamma_{\mathsf{ppt}}$, $\text{IDEAL}\langle \mathsf{Test} \mid \mathbf{\Sigma} \mid \mathsf{User} \rangle \approx \text{IDEAL}\langle \mathsf{Test} \circ \mathcal{O} \mid \mathbf{\Sigma}^* \mid \mathcal{E} \circ \mathsf{User} \rangle$.
2. *Simulation:* $\forall \mathsf{Test} \in \Gamma$, $\text{IDEAL}\langle \mathsf{Test} \mid \mathbf{\Sigma} \mid \mathcal{S} \circ \mathsf{User} \rangle \approx \text{IDEAL}\langle \mathsf{Test} \circ \mathcal{O} \mid \mathbf{\Sigma}^* \mid \mathsf{User} \rangle$.

If $\Gamma = \Gamma_{\mathsf{ppt}}$*, we simply say* Π *reduces* $\mathbf{\Sigma}$ *to* $\mathbf{\Sigma}^*$*. If there exists a scheme that reduces* $\mathbf{\Sigma}$ *to* $\mathbf{\Sigma}^*$*, then we say* $\mathbf{\Sigma}$ *reduces to* $\mathbf{\Sigma}^*$*. (Note that correctness is required for all* PPT Test*, and not just in* Γ*.)*

Figure 2 illustrates a reduction. It also shows how such a reduction can be composed with an $\mathsf{IND\text{-}PRE}$-secure scheme for $\mathbf{\Sigma}^*$. Below, we shall use $(\mathcal{O}', \mathcal{E}') = (\mathcal{O} \circ \mathcal{O}^*, \mathcal{E}^* \circ \mathcal{E})$ to denote the composed scheme in Figure 2(c).[9]

Theorem 1 (Composition)**.** *For any two schemata, $\mathbf{\Sigma}$ and $\mathbf{\Sigma}^*$, if $(\mathcal{O}, \mathcal{E})$ reduces $\mathbf{\Sigma}$ to $\mathbf{\Sigma}^*$ and $(\mathcal{O}^*, \mathcal{E}^*)$ is an $\mathsf{IND\text{-}PRE}$ secure scheme for $\mathbf{\Sigma}^*$, then $(\mathcal{O} \circ \mathcal{O}^*, \mathcal{E}^* \circ \mathcal{E})$ is an $\mathsf{IND\text{-}PRE}$ secure scheme for $\mathbf{\Sigma}$.*

Proof sketch: Let $(\mathcal{O}', \mathcal{E}') = (\mathcal{O} \circ \mathcal{O}^*, \mathcal{E}^* \circ \mathcal{E})$. Also, let $\mathsf{Test}' = \mathsf{Test} \circ \mathcal{O}$ and $\mathsf{User}' = \mathcal{E} \circ \mathsf{User}$. To show correctness, note that for any User, we have

$$\begin{aligned}\text{REAL}\langle \mathsf{Test} \mid \mathcal{O}' \mid \mathcal{E}' \circ \mathsf{User}\rangle = \text{REAL}\langle \mathsf{Test}' \mid \mathcal{O}^* \mid \mathcal{E}^* \circ \mathsf{User}'\rangle &\overset{(a)}{\approx} \text{IDEAL}\langle \mathsf{Test}' \mid \mathbf{\Sigma}^* \mid \mathsf{User}'\rangle \\ = \text{IDEAL}\langle \mathsf{Test} \circ \mathcal{O} \mid \mathbf{\Sigma}^* \mid \mathcal{E} \circ \mathsf{User}\rangle &\overset{(b)}{\approx} \text{IDEAL}\langle \mathsf{Test} \mid \mathbf{\Sigma} \mid \mathsf{User}\rangle\end{aligned}$$

where (a) follows from the correctness guarantee of $\mathsf{IND\text{-}PRE}$ security of $(\mathcal{O}^*, \mathcal{E}^*)$, and (b) follows from the correctness guarantee of $(\mathcal{O}, \mathcal{E})$ being a reduction of $\mathbf{\Sigma}$ to $\mathbf{\Sigma}^*$. (The other equalities are by regrouping the components in the system.)

It remains to prove that $\forall$ PPT Test, Test is hiding w.r.t. $\mathbf{\Sigma} \Rightarrow \mathsf{Test}$ is hiding w.r.t. $\mathcal{O}'$.

Firstly, we argue that Test is hiding w.r.t. $\mathbf{\Sigma} \Rightarrow \mathsf{Test}'$ is hiding w.r.t. $\mathbf{\Sigma}^*$. Suppose Test' is not hiding w.r.t. $\mathbf{\Sigma}^*$; then there is some User such that $\text{IDEAL}\langle \mathsf{Test}'(0) \mid \mathbf{\Sigma}^* \mid \mathsf{User}\rangle \not\approx \text{IDEAL}\langle \mathsf{Test}'(1) \mid \mathbf{\Sigma}^* \mid \mathsf{User}\rangle$. But, by security of the reduction $(\mathcal{O}, \mathcal{E})$ of $\mathbf{\Sigma}$ to $\mathbf{\Sigma}^*$, $\text{IDEAL}\langle \mathsf{Test}'(b) \mid \mathbf{\Sigma}^* \mid \mathsf{User}\rangle \approx \text{IDEAL}\langle \mathsf{Test}(b) \mid \mathbf{\Sigma} \mid \mathcal{S} \circ \mathsf{User}\rangle$, for $b = 0, 1$. Then, $\text{IDEAL}\langle \mathsf{Test}(0) \mid \mathbf{\Sigma} \mid \mathcal{S} \circ \mathsf{User}\rangle \not\approx \text{IDEAL}\langle \mathsf{Test}(1) \mid \mathbf{\Sigma} \mid \mathcal{S} \circ \mathsf{User}\rangle$, showing that Test is not hiding w.r.t. $\mathbf{\Sigma}$. Thus we have,

$$\begin{aligned}\mathsf{Test} \text{ is hiding w.r.t. } \mathbf{\Sigma} &\Rightarrow \mathsf{Test}' \text{ is hiding w.r.t. } \mathbf{\Sigma}^* \\ &\Rightarrow \mathsf{Test}' \text{ is hiding w.r.t. } \mathcal{O}^* \quad \text{since } (\mathcal{O}^*, \mathcal{E}^*) \ \mathsf{IND\text{-}PRE} \text{ securely implements } \mathbf{\Sigma}^* \\ &\Rightarrow \mathsf{Test} \text{ is hiding w.r.t. } \mathcal{O}'\end{aligned}$$

where the last implication follows by observing that for any Adv, we have $\text{REAL}\langle \mathsf{Test}' \mid \mathcal{O}^* \mid \mathsf{Adv}\rangle = \text{REAL}\langle \mathsf{Test} \mid \mathcal{O}' \mid \mathsf{Adv}\rangle$ (by regrouping the components). ☐

Note that in the above proof, we invoked the security guarantee of $(\mathcal{O}^*, \mathcal{E}^*)$ only with respect to tests of the form $\mathsf{Test} \circ \mathcal{O}$. Let $\Gamma \circ \mathcal{O} = \{\mathsf{Test} \circ \mathcal{O} | \mathsf{Test} \in \Gamma\}$. Then we have the following generalization.

Theorem 2 (Generalized Composition)**.** *For any two schemata, $\mathbf{\Sigma}$ and $\mathbf{\Sigma}^*$, if $(\mathcal{O}, \mathcal{E})$ reduces $\mathbf{\Sigma}$ to $\mathbf{\Sigma}^*$ and $(\mathcal{O}^*, \mathcal{E}^*)$ is a $(\Gamma \circ \mathcal{O})$-$\mathsf{IND\text{-}PRE}$ secure scheme for $\mathbf{\Sigma}^*$, then $(\mathcal{O} \circ \mathcal{O}^*, \mathcal{E}^* \circ \mathcal{E})$ is a Γ-$\mathsf{IND\text{-}PRE}$ secure scheme for $\mathbf{\Sigma}$.*

Theorem 3 (Transitivity of Reduction)**.** *For any three schemata, $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2, \mathbf{\Sigma}_3$, if $\mathbf{\Sigma}_1$ reduces to $\mathbf{\Sigma}_2$ and $\mathbf{\Sigma}_2$ reduces to $\mathbf{\Sigma}_3$, then $\mathbf{\Sigma}_1$ reduces to $\mathbf{\Sigma}_3$.*

Proof sketch: If $\Pi_1 = (\mathcal{O}_1, \mathcal{E}_1)$ and $\Pi_2 = (\mathcal{O}_2, \mathcal{E}_2)$ are schemes that carry out the Δ-reduction of $\mathbf{\Sigma}_1$ to $\mathbf{\Sigma}_2$ and that of $\mathbf{\Sigma}_2$ to $\mathbf{\Sigma}_3$, respectively, we claim that the scheme $\Pi = (\mathcal{O}_1 \circ \mathcal{O}_2, \mathcal{E}_2 \circ \mathcal{E}_1)$ is a reduction of $\mathbf{\Sigma}_1$ to $\mathbf{\Sigma}_3$. The correctness of this reduction follows from the correctness of the given reductions. Further, if $\mathcal{S}_1$ and $\mathcal{S}_2$ are the simulators associated with the two reductions, we can define a simulator $\mathcal{S}$ for the composed reduction as $\mathcal{S}_2 \circ \mathcal{S}_1$. ☐

[9] If $(\mathcal{O}, \mathcal{E})$ and $(\mathcal{O}^*, \mathcal{E}^*)$ have a setup phase, then it is implied that $\mathcal{O}'_{\mathsf{auth}} = \mathcal{O}_{\mathsf{auth}} \circ \mathcal{O}^*_{\mathsf{auth}}$, $\mathcal{O}'_{\mathsf{user}} = \mathcal{O}_{\mathsf{user}} \circ \mathcal{O}^*_{\mathsf{user}}$; invoking $\mathcal{O}'_{\mathsf{setup}}$ invokes both $\mathcal{O}_{\mathsf{setup}}$ and $\mathcal{O}^*_{\mathsf{setup}}$, and may in addition invoke $\mathcal{O}^*_{\mathsf{auth}}$ or $\mathcal{O}^*_{\mathsf{user}}$.

5 Restricted Test Families: Δ, Δ^* and Δ_{det}

The cryptographic agents framework is general enough to capture several important primitives such as obfuscation, functional encryption, fully homomorphic encryption and the like. However, as we will show in Section 7.3, for some schemata of interest, such as obfuscation (and hence function-hiding public key functional encryption which implies obfuscation) there exist no $\mathsf{IND\text{-}PRE}$ secure schemes. This motivates considering security against restricted families of tests which bypass these impossibilities. We remark that it would be possible to consider various families adapted to the existing security definitions of various primitives, but our goal is to provide a general framework that applies meaningfully to all primitives, and also, equipped with a composable notion of reduction. Towards this we propose the following sub-class of PPT tests, called Δ.

Informally, $\mathsf{Test} \in \Delta$ has the following form: each time Test sends an agent to $\mathcal{B}[\mathbf{\Sigma}]$, it picks two agents (P_0, P_1). Both the agents are sent to User, and P_b is sent to $\mathcal{B}[\mathbf{\Sigma}]$ (where b is the secret bit input to Test). Except for selecting the agent to be sent to $\mathcal{B}[\mathbf{\Sigma}]$, Test is oblivious to the bit b. It will be convenient to represent $\mathsf{Test}(b)$ (for $b \in \{0, 1\}$) as $\mathsf{D} \circ \mathfrak{c} \circ \mathfrak{s}(b)$, where D is a PPT party which communicates with User, and outputs pairs of the form (P_0, P_1) to $\mathfrak{c}$; $\mathfrak{c}$ sends both the agents to User, and also forwards them to $\mathfrak{s}$; $\mathfrak{s}(b)$ forwards P_b to $\mathcal{B}[\mathbf{\Sigma}]$ (and sends nothing to User).

As we shall see, for both obfuscation and functional encryption, Δ-$\mathsf{IND\text{-}PRE}$-security is indeed stronger than all the standard indistinguishability based security definitions in the literature.

But a drawback of restricting to a strict subset of all PPT tests is that the composition theorems (Theorem 1 and Theorem 3) do not hold any more. This is because, these composition theorems crucially relied on being able to define $\mathsf{Test}' = \mathsf{Test} \circ \mathcal{O}$ as a member of the test family, where $\mathcal{O}$ was defined by the reduction (see Theorem 2). Nevertheless, as we shall see, analogous composition theorems do exist for Δ, if we enhance the definition of a reduction. At a high-level, we shall require $\mathcal{O}$ to have some natural additional properties that would let us convert $\mathsf{Test} \circ \mathcal{O}$ back to a test in Δ, if Test itself belongs to Δ.

Combining Machines: Some Notation. Before defining Δ-reduction and proving the related composition theorems, it will be convenient to introduce some additional notation. Note that the machines $\mathfrak{c}$ and $\mathfrak{s}$ above, as well as the program $\mathcal{O}$, have three communication ports (in addition to the secret bit that $\mathfrak{s}$ receives): in terms of Figure 3, there is an input port below, an output port above and another output port on the right, to communicate with User. (D is also similar, except that it has no input port below, and on the right, it can interact with User by sending and receiving messages.) For such machines, we use $M_1 \circ M_2$ to denote connecting the output port above M_1 to the input port of M_2. The message from $M_1 \circ M_2$ to User is defined to consist of the pair of messages from M_1 and M_2 (formatted into a single message).

We shall also consider adding machines to the right of such a machine. Specifically, we use $M \,/\, K$ to denote modifying M using a machine K that takes as input the messages output by M to User (i.e., to its right), and to each such message may *append* an additional message of its own.

Also, recall that for two systems M and M', we say $M \approxeq M'$ if the two systems are indistinguishable to an interactive PPT distinguisher.

Using this notation, we define Δ-reduction.

Definition 8 (Δ-Reduction). *We say that a (hybrid) obfuscated agent scheme $\Pi = (\mathcal{O}, \mathcal{E})$ Δ-reduces $\mathbf{\Sigma}$ to $\mathbf{\Sigma}^*$ if*

1. *Π reduces $\mathbf{\Sigma}$ to $\mathbf{\Sigma}^*$ with respect to Δ (as in Definition 7), and*
2. *there exists PPT H and K such that*
   (a) *for all D such that $\mathsf{D} \circ \mathfrak{c} \circ \mathfrak{s}$ is hiding w.r.t. $\mathbf{\Sigma}$, $\mathsf{D} \circ \mathfrak{c} \circ \mathfrak{s}(b) \circ \mathcal{O} \approxeq \mathsf{D} \circ \mathfrak{c} \circ \mathsf{H} \circ \mathfrak{s}(b)$, for $b \in \{0, 1\}$;*

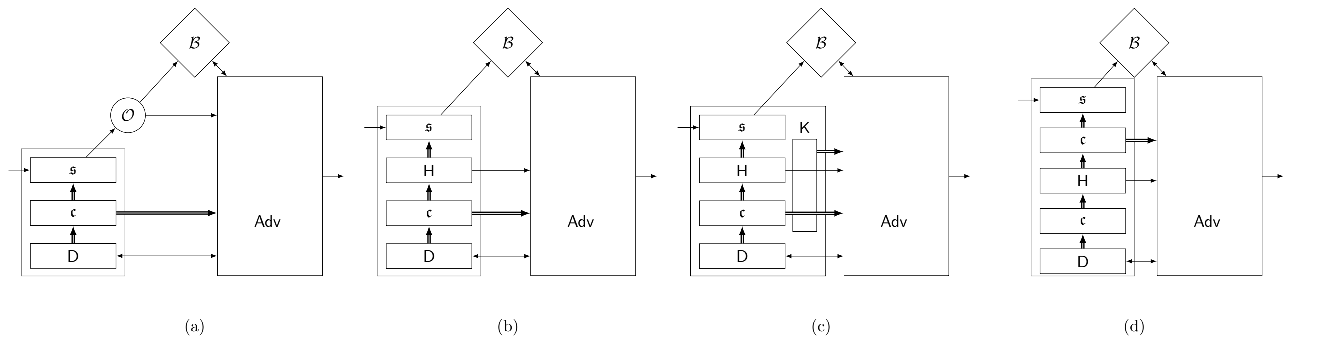

Figure 3 Illustration of Δ and the extra requirements on Δ-reduction. (a) illustrates the structure of a test in Δ; the double-arrows indicate messages consisting of a pair of agents. The first condition on H is that (a) and (b) are indistinguishable to Adv: i.e., H can mimic the message from $\mathcal{O}$ without knowing the input to $\mathfrak{s}$. The second condition is that (c) and (d) are indistinguishable: i.e., K should be able to simulate the pairs of agents produced by H based on the input to H (copied by $\mathfrak{c}$ to Adv) and the messages from H to Adv.

(b) $\mathfrak{c} \circ \mathsf{H} \circ \mathfrak{c} \approx \mathfrak{c} \circ \mathsf{H} \,/\, \mathsf{K}$.

If there exists a scheme that Δ-reduces $\mathbf{\Sigma}$ to $\mathbf{\Sigma}^*$, then we say $\mathbf{\Sigma}$ Δ-reduces to $\mathbf{\Sigma}^*$.

Informally, condition (a) allows us to move $\mathcal{O}$ "below" $\mathfrak{s}(b)$: note that H will need to send any messages $\mathcal{O}$ used to send to User, without knowing b. Condition (b) requires that sending a copy of the pairs of agents output by H (by adding $\mathfrak{c}$ "above" H) is "safe": it can be simulated by K, which only sees the pair of agents that are given as input to H.

Theorem 4 (Δ-Composition)**.** *For any two schemata, $\mathbf{\Sigma}$ and $\mathbf{\Sigma}^*$, if $(\mathcal{O}, \mathcal{E})$ Δ-reduces $\mathbf{\Sigma}$ to $\mathbf{\Sigma}^*$ and $(\mathcal{O}^*, \mathcal{E}^*)$ is a Δ-IND-PRE secure implementation of $\mathbf{\Sigma}^*$, then $(\mathcal{O} \circ \mathcal{O}^*, \mathcal{E}^* \circ \mathcal{E})$ is a Δ-IND-PRE secure implementation of $\mathbf{\Sigma}$.*

Proof sketch: Correctness and efficiency are easily confirmed. To prove security, we need to show that for every $\mathsf{Test} \in \Delta$, if Test is hiding w.r.t. $\mathbf{\Sigma}$, then it is hiding w.r.t. $\mathcal{O} \circ \mathcal{O}^*$. Since $\mathsf{Test} \in \Delta$, we can write it as $\mathsf{D} \circ \mathfrak{c} \circ \mathfrak{s}$. Let $\mathsf{Test}' \in \Delta$ be defined as $\mathsf{D} \circ \mathfrak{c} \circ \mathsf{H} \circ \mathfrak{c} \circ \mathfrak{s}$, where H is relates to $\mathcal{O}$ as in Definition 8.

First we shall argue that Test' is hiding w.r.t. $\mathbf{\Sigma}^*$. Below, we shall also use K that relates to H as in Definition 8. For ay PPT User, for each $b \in \{0,1\}$, we have

$$\begin{aligned} \mathsf{Test}'(b) &\equiv \mathsf{D} \circ \mathfrak{c} \circ \mathsf{H} \circ \mathfrak{c} \circ \mathfrak{s}(b) \\ &\approx \mathsf{D} \circ \mathfrak{c} \circ \mathsf{H} \circ \mathfrak{s}(b) \,/\, \mathsf{K} \\ &\approx \mathsf{D} \circ \mathfrak{c} \circ \mathfrak{s}(b) \circ \mathcal{O} \,/\, \mathsf{K} \equiv \mathsf{Test}(b) \circ \mathcal{O} \,/\, \mathsf{K}. \end{aligned} \tag{1}$$

So for any PPT User,

$$\begin{aligned} \textsc{ideal}\langle \mathsf{Test}'(b) \mid \mathbf{\Sigma}^* \mid \mathsf{User} \rangle &\approx \textsc{ideal}\langle \mathsf{Test}(b) \circ \mathcal{O} \,/\, \mathsf{K} \mid \mathbf{\Sigma}^* \mid \mathsf{User} \rangle \\ &= \textsc{ideal}\langle \mathsf{Test}(b) \circ \mathcal{O} \mid \mathbf{\Sigma}^* \mid \mathsf{User}' \rangle \qquad \text{where } \mathsf{User}' \text{ incorporates } \mathsf{K} \text{ and } \mathsf{User} \\ &= \textsc{ideal}\langle \mathsf{Test}(b) \mid \mathbf{\Sigma} \mid \mathcal{S} \circ \mathsf{User}' \rangle \qquad \text{where } \mathcal{S} \text{ is from Definition 7.} \end{aligned}$$

Hence if Test is hiding w.r.t. $\mathbf{\Sigma}$, $\textsc{ideal}\langle \mathsf{Test}(0) \mid \mathbf{\Sigma} \mid \mathsf{User}'' \rangle \approx \textsc{ideal}\langle \mathsf{Test}(1) \mid \mathbf{\Sigma} \mid \mathsf{User}'' \rangle$, where User'' stands for $\mathcal{S} \circ \mathsf{User}'$, and hence $\textsc{ideal}\langle \mathsf{Test}'(0) \mid \mathbf{\Sigma}^* \mid \mathsf{User} \rangle \approx \textsc{ideal}\langle \mathsf{Test}'(1) \mid \mathbf{\Sigma}^* \mid \mathsf{User} \rangle$. Since this holds for all PPT User, Test' is hiding w.r.t. $\mathbf{\Sigma}^*$. Thus we have,

$$\begin{aligned} \mathsf{Test} \text{ is hiding w.r.t. } \mathbf{\Sigma} &\Rightarrow \mathsf{Test}' \text{ is hiding w.r.t. } \mathbf{\Sigma}^* \\ &\Rightarrow \mathsf{Test}' \text{ is hiding w.r.t. } \mathcal{O}^* \qquad \text{as } (\mathcal{O}^*, \mathcal{E}^*) \text{ IND-PRE securely implements } \mathbf{\Sigma}^* \\ &\Rightarrow \mathsf{Test} \circ \mathcal{O} \,/\, \mathsf{K} \text{ is hiding w.r.t. } \mathcal{O}^* \qquad \text{by Equation 1.} \end{aligned}$$

Now, since K only provides extra information to User, if $\mathsf{Test} \circ \mathcal{O} \ / \ \mathsf{K}$ is hiding w.r.t. $\mathcal{O}^*$, then $\mathsf{Test} \circ \mathcal{O}$ is hiding w.r.t. $\mathcal{O}^*$. This is the same as saying that $\mathsf{Test} \circ \mathcal{O} \circ \mathcal{O}^*$ is hiding (w.r.t. a null scheme), as was required to be shown. $\square$

Theorem 5 (Transitivity of Δ-Reduction)**.** *For any three schemata, $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2, \mathbf{\Sigma}_3$, if $\mathbf{\Sigma}_1$ Δ-reduces to $\mathbf{\Sigma}_2$ and $\mathbf{\Sigma}_2$ Δ-reduces to $\mathbf{\Sigma}_3$, then $\mathbf{\Sigma}_1$ Δ-reduces to $\mathbf{\Sigma}_3$.*

Proof sketch: Let $\Pi_1 = (\mathcal{O}_1, \mathcal{E}_1)$ and $\Pi_2 = (\mathcal{O}_2, \mathcal{E}_2)$ be the schemes that carry out the Δ-reduction of $\mathbf{\Sigma}_1$ to $\mathbf{\Sigma}_2$ and that of $\mathbf{\Sigma}_2$ to $\mathbf{\Sigma}_3$, respectively. We define the scheme $\Pi = (\mathcal{O}_1 \circ \mathcal{O}_2, \mathcal{E}_2 \circ \mathcal{E}_1)$. As in Theorem 3, we see that Π reduces $\mathbf{\Sigma}_1$ to $\mathbf{\Sigma}_3$ with respect to Δ. What remains to be shown is that Π also has associated machines (H, K) as required in Definition 8.

Let $(\mathsf{H}_1, \mathsf{K}_1)$ and $(\mathsf{H}_2, \mathsf{K}_2)$ be associated with Π_1 and Π_2 respectively, as in Definition 8. We let $\mathsf{H} \equiv \mathsf{H}_1 \circ \mathsf{H}_2$. To define K, consider the cascade $\mathsf{K}_1 \ / \ \mathsf{K}_2$: i.e., K_1 appends a message to the first part of the input to K (from $\mathfrak{c} \circ \mathsf{H}_1$) and passes it on to K_2, which also gets the second part of the input (from H_2), and appends another message of its own. K behaves as $\mathsf{K}_1 \ / \ \mathsf{K}_2$ but from the output, it removes the message added by K_1. We write this as $\mathsf{K} \equiv \mathsf{K}_1 \ / \ \mathsf{K}_2 \ /\!\!/\mathsf{trim}$, where $/\!\!/\mathsf{trim}$ stands for the operation of redacting the appropriate part of the message. Note that K has the required format, in that it only appends to the entire message it receives.

We confirm that (H, K) satisfy the two required properties:

$$\mathfrak{s}(b) \circ \mathcal{O} \equiv \mathfrak{s}(b) \circ \mathcal{O}_1 \circ \mathcal{O}_2 \approxeq \mathsf{H}_1 \circ \mathfrak{s}(b) \circ \mathcal{O}_2 \approxeq \mathsf{H}_1 \circ \mathsf{H}_2 \circ \mathfrak{s}(b) \equiv \mathsf{H} \circ \mathfrak{s}(b)$$
$$\mathfrak{c} \circ \mathsf{H} \ / \ \mathsf{K} \equiv (\mathfrak{c} \circ \mathsf{H}_1 \ / \ \mathsf{K}_1) \circ \mathsf{H}_2 \ / \ \mathsf{K}_2 \ /\!\!/\mathsf{trim} \ \approxeq \mathfrak{c} \circ \mathsf{H}_1 \circ \mathfrak{c} \circ \mathsf{H}_2 \ / \ \mathsf{K}_2 \ /\!\!/\mathsf{trim}$$
$$\approxeq \mathfrak{c} \circ \mathsf{H}_1 \circ \mathfrak{c} \circ \mathsf{H}_2 \circ \mathfrak{c} \ /\!\!/\mathsf{trim} \ \equiv \mathfrak{c} \circ \mathsf{H}_1 \circ \mathsf{H}_2 \circ \mathfrak{c}$$

where the last identity follows from the fact that the operation $/\!\!/\mathsf{trim}$ removes the appropriate part of the outgoing message. $\square$

Other Restricted Test Families. We define two more restricted test families, Δ^* and Δ_{det}, which are of great interest for the obfuscation and functional encryption schemata. Both of these are subsets of Δ.

Δ_{det} simply consists of all deterministic tests in Δ. Equivalently, Δ_{det} is the class of all tests of the form $\mathsf{D} \circ \mathfrak{c} \circ \mathfrak{s}$, where D is a deterministic polynomial time party which communicates with User, and outputs pairs of the form (P_0, P_1) to $\mathfrak{c}$.

Δ^* consists of all tests in Δ which do not read any messages from User. Equivalently, Δ^* is the class of all tests of the form $\mathsf{D} \circ \mathfrak{c} \circ \mathfrak{s}$, where D is a PPT party which may send messages to User but does not accept any messages from User, and outputs pairs of the form (P_0, P_1) to $\mathfrak{c}$.

The composition theorem for Δ, Theorem 4 holds for Δ^* as well, as can be seen from the proof above.

Theorem 6 (Δ^*-Composition)**.** *For any two schemata, $\mathbf{\Sigma}$ and $\mathbf{\Sigma}^*$, if $(\mathcal{O}, \mathcal{E})$ Δ-reduces $\mathbf{\Sigma}$ to $\mathbf{\Sigma}^*$ and $(\mathcal{O}^*, \mathcal{E}^*)$ is a Δ^*-IND-PRE secure implementation of $\mathbf{\Sigma}^*$, then $(\mathcal{O} \circ \mathcal{O}^*, \mathcal{E}^* \circ \mathcal{E})$ is a Δ^*-IND-PRE secure implementation of $\mathbf{\Sigma}$.*

Note that here the notion of reduction is still the same as in Theorem 4, namely Δ-reduction. (In contrast, this result does *not* extend to Δ_{det}, unless the notion of reduction is severely restricted, by requiring H and K to be deterministic.)

6 Generic Group Schema

Our framework provides a method to convert a certain class of constructions — i.e., secure schemes for primitives that can be modeled as schemata — that are proven secure in heuristic models like the random oracle model [14] or the (bilinear) generic group model [77, 69], into secure constructions in the standard model.

To be concrete, we consider the case of the generic group model. There are two important observations we make:

- Proving that a cryptographic scheme for a given schema $\mathbf{\Sigma}$ is secure in the generic group model typically amounts to a *reduction from* $\mathbf{\Sigma}$ *to a "generic group schema"* $\mathbf{\Sigma}_{\text{GG}}$.
- The assumption that there is an $\mathsf{IND\text{-}PRE}$-secure scheme Π_{GG} for $\mathbf{\Sigma}_{\text{GG}}$ is a standard-model assumption (that does not appear to be ruled out by known results or techniques).

Combined using the composition theorem (Theorem 1), these two observations yield a standard model construction for an $\mathsf{IND\text{-}PRE}$-secure scheme for $\mathbf{\Sigma}$.

Above, the generic group schema $\mathbf{\Sigma}_{\text{GG}}$ is defined in a natural way: the agents (all in $\mathcal{P}_{\mathsf{user}}$, with $\mathcal{P}_{\mathsf{auth}} = \emptyset$) are parametrized by elements of a large (say cyclic) group, and interact with each other to carry out group operations; the only output the agents produce for a user is the result of checking equality with another agent.

We formally state the assumption mentioned above:

Assumption 1 (Γ-Generic Group Agent Assumption)**.** *There exists a* Γ-$\mathsf{IND\text{-}PRE}$*-secure scheme for the generic group schema* $\mathbf{\Sigma}_{\text{GG}}$*.*

Similarly, we put forward the Γ-*Bilinear* Generic Group Agent Assumption, where $\mathbf{\Sigma}_{\text{GG}}$ is replaced by $\mathbf{\Sigma}_{\text{BGG}}$ which has three groups (two source groups and a target group), and allows the bilinear pairing operation as well.

The most useful form of these assumptions (required by the composition theorem when used with the standard reduction) is when Γ is the set of all PPT tests. However, weaker forms of this assumption (like Δ-GGA assumption, or Δ^*-GGA assumption) are also useful, if a given construction could be viewed as a stronger form of reduction (like Δ-reduction).

While this assumption may appear too strong at first sight – given the impossibility results surrounding the generic group model – we argue that it is plausible. Firstly, observe that primitives that can be captured as schemata are somewhat restricted: primitives like zero knowledge that involve simulation based security, CCA secure encryption or non-committing encryption and such others do not have an interpretation as a secure schema. Secondly, $\mathsf{IND\text{-}PRE}$ security is weaker than simulation based security, and its achievability is not easily ruled out (see discussion in Section 10). Also we note that such an assumption already exists in the context of another popular idealized model: the random oracle model (ROM). Specifically, consider a natural definition of the random oracle schema, $\mathbf{\Sigma}_{\text{RO}}$, in which the agents encode elements in a large set and interact with each other to carry out equality checks. Then, a Δ_{det}-$\mathsf{IND\text{-}PRE}$-secure scheme for $\mathbf{\Sigma}_{\text{RO}}$ is equivalent to a point obfuscation scheme, which hides everything about the input except the output. The assumption that such a scheme exists is widely considered plausible, and has been the subject of prior research [35, 39, 79, 37]. This fits into a broader theme of research that attempts to capture several features of the random oracle using standard model assumptions (e.g., [17, 12]). The GGA assumption above can be seen as a similar approach to the generic group model, that captures only some of the security guarantees of the generic group model so that it becomes a plausible assumption in the standard model, yet is general enough to be of use in a broad class of applications.

One may wonder if we could use an even stronger assumption, by replacing the (bilinear) generic group schema $\mathbf{\Sigma}_{\text{GG}}$ or $\mathbf{\Sigma}_{\text{BGG}}$ by a multi-linear generic group schema $\mathbf{\Sigma}_{\text{MGG}}$, which permits black box computation of multilinear map operations [22, 49]. Interestingly, this assumption is provably false, since there exists a reduction of obfuscation schema $\mathbf{\Sigma}_{\text{OBF}}$ to $\mathbf{\Sigma}_{\text{MGG}}$ [31, 8], and we have seen that there is no $\mathsf{IND\text{-}PRE}$-secure scheme for $\mathbf{\Sigma}_{\text{OBF}}$. We leave it as a fascinating question to understand the level of complexity of a schema that makes it unrealizable, irrespective of computational assumptions.

Falsifiability. Note that the above assumption as stated is not necessarily falsifiable, since there is no easy way to check that a given PPT test is hiding. However, it becomes falsifiable if instead of $\mathsf{IND\text{-}PRE}$ security, we used a modified notion of security $\mathsf{IND\text{-}PRE}'$, which requires that every test which is *efficiently provably* ideal-hiding is real-hiding. We note that $\mathsf{IND\text{-}PRE}'$ security suffices for all practical purposes as a security guarantee, and also suffices for the composition theorem. With this notion, to falsify the assumption, the adversary can (and must) provide a proof that a test is ideal-hiding and also exhibit a real world adversary who breaks its hiding when using the scheme.

7 Obfuscation Schema

In this section we define and study the obfuscation schema $\mathbf{\Sigma}_{\text{OBF}}$. In the obfuscation schema, agents are deterministic, non-interactive and non-reactive: such an agent behaves as a simple Turing machine, that reads an input, produces an output and halts. We begin by showing that the obfuscation schema is "complete" under reduction as defined in Definition 7. Thus, there is an $\mathsf{IND\text{-}PRE}$ secure implementation $(\mathcal{O}, \mathcal{E})$ which reduces any general $\mathbf{\Sigma}_F$ to $\mathbf{\Sigma}_{\text{OBF}}$. This means that if there is an $\mathsf{IND\text{-}PRE}$ secure implementation of $\mathbf{\Sigma}_{\text{OBF}}$, say using secure hardware, then this implementation can be used in a modular way to build an $\mathsf{IND\text{-}PRE}$ secure schema for any general functionality.

Next, we show there cannot exist an $\mathsf{IND\text{-}PRE}$ secure schema for general functionalities. Thus, we exhibit a class of programs $\mathcal{F}$ such that Test is hiding in the ideal world but for any real world cryptographic scheme $(\mathcal{O}, \mathcal{E})$, Test is not hiding in the real world. Our impossibility follows the broad outline of the impossibility of virtual black box obfuscation by Barak et al. [9]. Since our definition of obfuscation schema is implied by virtual black box obfuscation, we obtain a potential[10] strenghening of the result of Barak et al. [9].

Finally, we relate the notion of $\mathsf{IND\text{-}PRE}$ obfuscation to standard notions of obfuscation such as indistinguishability obfuscation and differing inputs obfuscation. We show that the former is equivalent to $\mathsf{IND\text{-}PRE}$ restricted to the Δ_{det} family of tests, while the latter is is equivalent to $\mathsf{IND\text{-}PRE}$ restricted to the Δ^* family of tests. We define a new notion of obfuscation corresponding to $\mathsf{IND\text{-}PRE}$ restricted to the Δ family of tests, namely *adaptive differing inputs obfuscation.*

7.1 Definition

Below, we formally define the obfuscation schema. If $\mathcal{F}$ is a family of deterministic, non-interactive and non-reactive agents, we define

$$\mathbf{\Sigma}_{\text{OBF}(\mathcal{F})} := (\emptyset, \mathcal{F}).$$

That is, in the ideal execution User obtains handles for computing $\mathcal{F}$. We shall consider setup-free, $\mathsf{IND\text{-}PRE}$ secure implementations $(\mathcal{O}, \mathcal{E})$ of $\mathbf{\Sigma}_{\text{OBF}(\mathcal{F})}$.

A special case of $\mathbf{\Sigma}_{\text{OBF}(\mathcal{F})}$ corresponds to the case when $\mathcal{F}$ is the class of all functions that can be computed within a certain amount of time. More precisely, we can define the agent family $\mathcal{U}_s$ (for *universal* computation) to consist of agents of the following form: the parameter tape, which is at most $s(\kappa)$ bits long is taken to contain (in addition to κ) the description of an arbitrary binary circuit C; on input x, $\mathcal{U}_s$ will compute and output $C(x)$ (padding or truncating x as necessary). We define the "general" obfuscation schema

$$\mathbf{\Sigma}_{\text{OBF}} := (\emptyset, \mathcal{P}^{\text{OBF}}_{\mathsf{user}}) := \mathbf{\Sigma}_{\text{OBF}(\mathcal{U}_s)},$$

[10] we do not have a separation between $\mathsf{IND\text{-}PRE}$ and VBB obfuscation so far

for a given polynomial s. Here we have omitted s from the notation $\mathbf{\Sigma}_{\text{OBF}}$ and $\mathcal{P}_{\mathsf{user}}^{\text{OBF}}$ for simplicity, but it is to be understood that whenever we refer to $\mathbf{\Sigma}_{\text{OBF}}$ some polynomial s is implied.

7.2 Completeness of Obfuscation

We show that $\mathbf{\Sigma}_{\text{OBF}}$ is a complete schema with respect to schematic reduction (Definition 7). That is, *every schema* (including possibly randomized, interactive, and stateful agents) can be reduced to $\mathbf{\Sigma}_{\text{OBF}}$. We stress that this does not yield an IND-PRE-secure scheme for every schema (using composition), since there does not exist an IND-PRE-secure scheme for $\mathbf{\Sigma}_{\text{OBF}}$, as described in Section 7.3.

The reduction uses only standard cryptographic primitives: CCA secure public-key encryption and digital signatures. We present the full construction and proof in Appendix B.

7.3 Impossibility of IND-PRE obfuscation for general functionalities

In this section we exhibit a class of programs $\mathcal{F}$ such that Test is hiding w.r.t $\mathbf{\Sigma}_{\text{OBF}(\mathcal{F})}$ but for any real world cryptographic scheme $(\mathcal{O}, \mathcal{E})$, Test is not hiding w.r.t. $\mathcal{O}$. The idea for our impossibility follows the broad outline of the impossibility of general virtual black box (VBB) obfuscation demonstrated by Barak et al. [9]. Intuitively the impossibility of VBB obfuscation by Barak et al. follows the following broad outline: consider a program P which expects code C as input. If the input code responds to a secret challenge α with a secret response β, then P outputs a secret bit b. Barak et al. show that using the code of P, one can construct nontrivial input code C that can be fed back to P forcing it to output the bit b. On the other hand, a simulator given oracle access to P cannot use it to construct a useful input code C and has negligible probability of guessing an input that will result in P outputting the secret b. For more details, we refer the reader to [9].

At first glance, it is not clear if the same argument can be used to rule out IND-PRE secure obfuscation schema. The argument by Barak et al. seems to rely crucially on simulation based security, whereas ours is an indistinguishability style definition. Indeed, other indistinguishability style definitions such as indistinguishability obfuscation (I-Obf) and differing input obfuscation (DI-Obf) are conjectured to exist for all functions. However, our notion of indistinguishability preserving obfuscation is too strong to be achieved, as we shall see. We will consider the same class of functions $\mathcal{F}$ as in [9], with the bit b as the secret. We construct Test which expects b as external input, and uploads agents from the function family $\mathcal{F}$. In the ideal world, it is infeasible to distinguish between Test(0) and Test(1) since it is infeasible to recover b from black box access. In the real world however, a user may execute a session in which the agent for P is executed to produce an agent for C, following which P may be run on C to output the secret bit. We defer the formal proof to the full version.

7.4 Relation to other notions of Obfuscation

In this section we relate the security notions for obfuscation schema to the standard notions of obfuscation known in literature, such as indistinguishability obfuscation and differing inputs obfuscation. These relaxations of the VBB definition were first proposed by Barak et al. [9, 10], but no constructions were discovered for a long time. Recently, Garg et al. [50] proposed the first candidate for indistinguishability obfuscation. Later works assume that this candidate is also a differing inputs obfuscator [7, 58].

7.4.1 Indistinguishability Obfuscation.

Loosely speaking, indistinguishability obfuscation posits that if two distinct circuits implement identical functionalities, then their obfuscations must be indistinguishable [9]. The formal definition is provided in Appendix C.

We show how a (set-up free) obfuscation scheme $(\mathcal{O}, \mathcal{E})$ in our framework can be converted to an indistinguishability obfuscator. First, it is easy to see that the efficiency requirement of $(\mathcal{O}, \mathcal{E})$ implies a pre-determined poly(κ) upperbound on the execution time of $\mathcal{E}$ on a single invocation (because, the agents in $\mathbf{\Sigma}_{\text{OBF}}$ have a pre-determined poly(κ) upperbound on their running time). Hence we can define a circuit $\mathcal{E}[O]$ (with a built-in string O) of a pre-determined poly(κ) size that carries out the following computation: on input x, it interacts with an internal copy of $\mathcal{E}$, first simulating to it the message O from $\mathcal{O}$ (upon which $\mathcal{E}$ will output a handle), followed by a request from User to execute a session with that handle and input x; $\mathcal{E}[O]$ outputs whatever $\mathcal{E}$ outputs. Let $\mathcal{O} \circ \mathcal{E}$ denote a program which, on input a circuit C (of size at most $s(\kappa)$), invokes $\mathcal{O}$ on C to obtain a string O, and then outputs the program $\mathcal{E}[O]$.

We now show that if we restrict our attention to the family of tests $\Delta_{\mathsf{det}} \subset \Delta$ where D is a deterministic party, then a secure scheme for this family exists iff an indistinguishability obfuscator does. Formally,

Lemma 1. *A set-up free Δ_{det}-$\mathsf{IND\text{-}PRE}$-secure scheme for $\mathbf{\Sigma}_{\text{OBF}}$ (with perfect correctness) exists if and only if there exists an indistinguishability obfuscator.*

The proof is provided in Appendix C.1.1.

7.4.2 Differing Inputs Obfuscation.

Next, we consider the notion of differing inputs obfuscation which strengthens the notion of indistinguishability obfuscation. At a high level, it stipulates that if the differing input obfuscations of two circuits $\mathsf{DI\text{-}Obf}(C_0)$ and $\mathsf{DI\text{-}Obf}(C_1)$ can be computationally distinguished, then there exists an adversary that can find an input on which the two circuits differ. Formal definition is provided in Appendix C.2. We show that if we consider the family of tests which do not receive any input from the user, then a secure scheme for this family exists iff a differing input obfuscator does. Formally,

Lemma 2. *A set-up free Δ^*-$\mathsf{IND\text{-}PRE}$-secure scheme for $\mathbf{\Sigma}_{\text{OBF}}$ (with perfect correctness) exists if and only if there exists a differing-inputs obfuscator.*

The proof is provided in Appendix C.2.1.

7.5 Adaptive Differing Inputs Obfuscation

In the previous section, we saw that indistinguishability obfuscation is equivalent to Δ_{det}-$\mathsf{IND\text{-}PRE}$ and differing inputs obfuscation is equivalent to Δ^*-$\mathsf{IND\text{-}PRE}$. In Section 7.3, we saw that $\mathsf{IND\text{-}PRE}$ secure obfuscation is impossible for general functionalities. It is natural to ask what happens "in-between", i.e. for Δ family of tests?

To this end, we state a new definition for the security of obfuscation – adaptive differing-inputs obfuscation, which is equivalent Δ-$\mathsf{IND\text{-}PRE}$ security. Informally, it is the same as differing inputs obfuscation, but an adversary is allowed to interact with the sampler (which samples two circuits one of which will be obfuscated and presented to the adversary as a challenge), even after it receives the obfuscation. We define it formally below.

Good sampler : Let $\mathcal{F} = \{\mathcal{F}_\kappa\}$ be a circuit family. Let $\mathsf{Sampler}$ be a PPT stateful oracle which takes 1^κ as input, and upon every invocation outputs two circuits $C_0, C_1 \in \mathcal{F}_\kappa$ and some auxiliary information aux. We call this oracle *good* if for every PPT adversary $\mathcal{A}$ with oracle access to $\mathsf{Sampler}$, the probability that $\mathcal{A}$ outputs an x such that $C_0(x) \neq C_1(x)$ for some C_0, C_1 given by $\mathsf{Sampler}$, is negligible in κ.

Definition 9 (Adaptive Differing Inputs Obfuscation)**.** *A uniform* PPT *machine* $\text{OBF}(\cdot)$ *is called an adaptive differing inputs obfuscator for a circuit family $\mathcal{F} = \{\mathcal{F}_\kappa\}$ if it probabilistically maps circuits to circuits such that it satisfies the following conditions:*

- **Correctness:** $\forall \kappa \in \mathbb{N}$, $\forall C \in \mathcal{F}_\kappa$, *and* $\forall$ *inputs* x *we have that*

$$\Pr\left[C'(x) = C(x) : C' \leftarrow \textsc{obf}(1^\kappa, C)\right] = 1.$$

- **Relaxed Polynomial Slowdown:** *There exists a universal polynomial* p *such that for any circuit* C*, we have* $|C'| \leq p(|C|, \kappa)$ *where* $C' \leftarrow \textsc{obf}(1^\kappa, C)$*.*
- **Adaptive Indistinguishability:** *Let* $\mathsf{Sampler}$ *be a stateful oracle as described above. Define* $\mathsf{Sampler}_b$ *to be an oracle that simulates* $\mathsf{Sampler}$ *internally, and when* $\mathsf{Sampler}$ *outputs* C_0, C_1 *and* aux*,* $\mathsf{Sampler}_b$ *additionally outputs* $\textsc{obf}(1^\kappa, C_b)$*. We require that for every good* $\mathsf{Sampler}$*, for all* PPT *distinguishers* $\mathcal{D}$

$$\mathcal{D}^{\mathsf{Sampler}_0}(1^\kappa) \approx \mathcal{D}^{\mathsf{Sampler}_1}(1^\kappa).$$

As we shall see, this notion of obfuscation is very useful and we will be able to construct Δ-IND-PRE FE schema by providing a Δ reduction to a Δ-IND-PRE secure obfuscation schema (see Section 8 for more details).

8 Functional Encryption

In this section, we present a schema $\mathbf{\Sigma}_{\text{FE}}$ for Functional Encryption. Although many variants of FE are known – public or secret key, with or without function hiding, indistinguishability based or simulation based – they can all[11] be captured as schemata secure against different families of test programs. For concreteness, we focus on adaptive secure, indistinguishability-based, public-key FE (with and without function-hiding). In Section 8.1 we introduce the schema $\mathbf{\Sigma}_{\text{FE}}$ for FE without function-hiding, and in Section 8.2 we introduce the schema $\mathbf{\Sigma}_{\text{FH-FE}}$ for function-hiding FE.

8.1 Functional Encryption without Function Hiding

Public-key FE without function-hiding is the most well-studied variant of FE.

8.1.1 Definition

For a circuit family $\mathcal{C} = \{\mathcal{C}_\kappa\}$ and a message space $\mathcal{X} = \{\mathcal{X}_\kappa\}$, we define the schema $\mathbf{\Sigma}_{\text{FE}} = (\mathcal{P}^{\text{FE}}_{\mathsf{auth}}, \mathcal{P}^{\text{FE}}_{\mathsf{user}})$ as follows:

- $\mathcal{P}^{\text{FE}}_{\mathsf{user}}$: An agent $P_x \in \mathcal{P}^{\text{FE}}_{\mathsf{user}}$ simply sends x to the first agent in the session, where $x \in \mathcal{X}$ is a parameter of the agent, and halts. We will often refer to such an agent as a *message agent*.
- $\mathcal{P}^{\text{FE}}_{\mathsf{auth}}$: An agent $P_C \in \mathcal{P}^{\text{FE}}_{\mathsf{auth}}$, when invoked with input 0, outputs C (where $C \in \mathcal{C}$ is a parameter of the agent) and halts. If invoked with input 1, it reads a message $\tilde{x}$ from its incoming communication tape, writes $C(\tilde{x})$ on its output tape and halts. We will often refer to such an agent as a *function agent*.

8.1.2 Reducing Functional Encryption to Obfuscation

In a sequence of recent results [50, 7, 1, 58, 25, 26], it was shown how to obtain various flavors of functional encryption from various flavors of obfuscation. We investigate this connection in terms of schematic reducibility: can $\mathbf{\Sigma}_{\text{FE}}$ be reduced to $\mathbf{\Sigma}_{\text{OBF}}$? For this reduction to translate to an IND-PRE-secure scheme for $\mathbf{\Sigma}_{\text{FE}}$, we will need (1) an IND-PRE-secure scheme for $\mathbf{\Sigma}_{\text{OBF}}$, and (2) a composition theorem.

Our main result in this section is a Δ-reduction of $\mathbf{\Sigma}_{\text{FE}}$ to $\mathbf{\Sigma}_{\text{OBF}}$. Then, combined with a Γ-IND-PRE secure implementation of $\mathbf{\Sigma}_{\text{OBF}}$, we obtain a Γ-IND-PRE secure implementation of $\mathbf{\Sigma}_{\text{FE}}$, where Γ is Δ (respectively, Δ^*), thanks to Theorem 4 (respectively, Theorem 6).

[11] Simulation-based definitions can be captured in terms of reduction to the null schema.

Before explaining our reduction, we compare it with the results in [50, 7, 25]. At a high-level, these works could be seen as giving $(\Gamma_{\text{FE}}, \Gamma_{\text{OBF}})$-reductions from $\mathbf{\Sigma}_{\text{FE}}$ to $\mathbf{\Sigma}_{\text{OBF}}$, which when composed with a Γ_{OBF}-$\mathsf{IND\text{-}PRE}$-secure scheme for $\mathbf{\Sigma}_{\text{OBF}}$ yields a Γ_{FE}-$\mathsf{IND\text{-}PRE}$-secure scheme for $\mathbf{\Sigma}_{\text{FE}}$. The obfuscation scheme in [50] was proposed as a candidate for a Δ_{det}-$\mathsf{IND\text{-}PRE}$-secure scheme for $\mathbf{\Sigma}_{\text{OBF}}$ (i.e., an indistinguishability obfuscation scheme). They also consider a *selective-secure* functional encryption which can be captured using a family of tests we term Δ_{sel}. That is, in [50], we have $(\Gamma_{\text{FE}}, \Gamma_{\text{OBF}}) = (\Delta_{\mathsf{sel}}, \Delta_{\mathsf{det}})$. Indeed, similarly to how we defined Δ-reduction, one can define an abstract notion of $(\Delta_{\mathsf{sel}}, \Delta_{\mathsf{det}})$-reduction (between any two schemata) with a corresponding composition theorem, and the reduction in [50] would fit this definition.

In [7, 25], the obfuscation scheme is considered to be a Δ^*-$\mathsf{IND\text{-}PRE}$-secure scheme for $\mathbf{\Sigma}_{\text{OBF}}$ (i.e., a differing-inputs obfuscation). The notion of functional encryption considered corresponds to a Δ_{det}-$\mathsf{IND\text{-}PRE}$-secure scheme for $\mathbf{\Sigma}_{\text{FE}}$. However, the reduction in [7, 25] will not satisfy a natural definition of $(\Delta_{\mathsf{det}}, \Delta^*)$-reduction. Instead, we can consider their reductions as (Γ, Δ^*)-reductions where Γ is a restriction of Δ_{det} that suffices to capture functional encryption. In this work we omit formal descriptions of Δ_{sel} and Γ above, as well as the definitions of $(\Delta_{\mathsf{sel}}, \Delta_{\mathsf{det}})$-reduction and (Γ, Δ^*)-reduction mentioned above, that can be used to capture the constructions in [50, 7, 25]. Instead, we propose Δ-$\mathsf{IND\text{-}PRE}$-security as a natural security notion for both obfuscation and functional encryption schemata, and provide a simpler Δ-reduction from $\mathbf{\Sigma}_{\text{FE}}$ to $\mathbf{\Sigma}_{\text{OBF}}$.

Our Construction. We shall use a simple and natural functional encryption scheme: the key for a function f is simply a description of f with a signature on it; a ciphertext of a message m is an obfuscation of a program which when given as input a signed description of a function f, returns $f(m)$ if the signature verifies (and $\perp$ otherwise). Essentially the same construction was used in [25] as well, but they rely on "functional signatures" in which it is possible to derive keys for signing only messages satisfying an arbitrary relation. In our construction, we need only a standard digital signature scheme.

Below we describe our construction more formally, as a reduction from $\mathbf{\Sigma}_{\text{FE}}$ to $\mathbf{\Sigma}_{\text{OBF}}$ and prove that it is in fact a Δ-reduction. Let $\mathbf{\Sigma}_{\text{FE}} = (\mathcal{P}^{\text{FE}}_{\mathsf{auth}}, \mathcal{P}^{\text{FE}}_{\mathsf{user}})$ and $\mathbf{\Sigma}_{\text{OBF}} = (\emptyset, \mathcal{P}^{\text{OBF}}_{\mathsf{user}})$.

We shall only describe $\mathcal{O} = (\mathcal{O}_{\mathsf{setup}}, \mathcal{O}_{\mathsf{auth}}, \mathcal{O}_{\mathsf{user}})$; $\mathcal{E}$ is naturally defined, and the correctness will be easy to verify.

- $\mathcal{O}_{\mathsf{setup}}$ picks a pair of signing and verification keys $(\mathsf{SK}, \mathsf{VK})$ for the signature scheme as $(\mathsf{MSK}, \mathsf{MPK})$.
- $\mathcal{O}_{\mathsf{auth}}$, when given a function agent $P_f \in \mathcal{P}^{\text{FE}}_{\mathsf{auth}}$, outputs (f, σ) to be sent to $\mathcal{E}$, where f is the parameter of P_f and σ is a signature on it.
- $\mathcal{O}_{\mathsf{user}}$, when given an agent $P_m \in \mathcal{P}^{\text{FE}}_{\mathsf{user}}$ as input, uploads an agent $P_{m,\mathsf{MPK}} \in \mathcal{P}^{\text{OBF}}_{\mathsf{user}}$ to $\mathcal{B}[\mathbf{\Sigma}_{\text{OBF}}]$, which behaves as follows: on input (f, σ) $P_{m,\mathsf{MPK}}$ verifies that σ is a valid signature on f with respect to the signature verification key MPK; if so, it outputs $f(m)$, and else $\perp$.

To show that this is a valid Δ-reduction, apart from verifying correctness, we need to demonstrate $\mathcal{S}$, H and K as required in Definition 7 and Definition 8. We describe these below.

- $\mathcal{S}$ will first simulate $\mathcal{O}_{\mathsf{setup}}$, by picking a signing and verification key pair itself, and publishing the latter as MPK. On obtaining a handle h_f for an agent in $\mathcal{P}^{\text{FE}}_{\mathsf{auth}}$, it runs the agent with no input to recover f, and then simulates $\mathcal{O}_{\mathsf{auth}}$ by outputting (f, σ) where σ is a signature on f. On obtaining a handle h for an agent in $\mathcal{P}^{\text{FE}}_{\mathsf{user}}$, it outputs a simulated handle h' from $\mathcal{B}[\mathbf{\Sigma}_{\text{OBF}}]$ (for the agent P_m uploaded by $\mathcal{O}_{\mathsf{user}}$), and internally keeps a record of the pair (h, h'). Subsequently, on receiving a session execution request for a simulated handle h' with some input, first $\mathcal{S}$ checks if the input is of the form (f, σ) and σ is a valid signature on f. If so, it looks for a handle h_f corresponding to f that it received from $\mathcal{B}[\mathbf{\Sigma}_{\text{FE}}]$; if no such handle exists, it aborts the simulation. Else it requests an execution of $\mathcal{B}[\mathbf{\Sigma}_{\text{FE}}]$ session involving two handles h_f and h, where (h, h') was the pair it had recorded when issuing the simulated handle h'. It returns the output from this $\mathcal{B}[\mathbf{\Sigma}_{\text{FE}}]$ session as the outcome of the execution of the simulated $\mathcal{B}[\mathbf{\Sigma}_{\text{OBF}}]$ session.

The probability $\mathcal{S}$ aborts is negligible, since any PPT adversary will have negligible probability of producing an f with a valid signature, if it was not given out by $\mathcal{S}$. Conditioned on the adversary never creating a forged signature, the simulated and real executions are identical.

- We can define H as follows. It implements $\mathcal{O}_{\mathsf{setup}}$ faithfully. When it is given a pair of agents in $\mathcal{P}^{\text{FE}}_{\mathsf{user}}$, it simply forwards both of them (to $\mathfrak{s}$). When it receives a pair of agents in $\mathcal{P}^{\text{FE}}_{\mathsf{auth}}$ from T, if they are not identical, H aborts; otherwise H will simulate the effect of $\mathcal{O}_{\mathsf{auth}}$ by signing the function f in (both) the agents, and forwards it to User. Now, conditioned on D never outputting a pair of distinct agents in $\mathcal{P}^{\text{FE}}_{\mathsf{auth}}$, we have $\mathsf{D} \circ \mathfrak{c} \circ \mathsf{H} \circ \mathfrak{s} \equiv \mathsf{D} \circ \mathfrak{c} \circ \mathfrak{s} \circ \mathcal{O}$.

Now, if $\mathsf{D} \circ \mathfrak{c} \circ \mathfrak{s}$ is hiding w.r.t. $\mathbf{\Sigma}_{\text{FE}}$, then it must be the case that the probability of D outputting a pair of distinct agents is negligible. This is because, $\mathsf{D} \circ \mathfrak{c} \circ \mathfrak{s}$ will forward the two agents to User, and if the two agents are not identical, the function-revealing nature of the schema, will let the User learn the secret bit b.

- We define K as follows. It observes the inputs sent to H (as reported to User by $\mathfrak{c}$), and whenever it sees a pair of agents in $\mathcal{P}^{\text{FE}}_{\mathsf{user}}$, it appends a copy of those two agents (as if it was reported the second instance of $\mathfrak{c}$ in $\mathfrak{c} \circ \mathsf{H} \circ \mathfrak{c}$). This in fact ensures that $\mathfrak{c} \circ \mathsf{H} \circ \mathfrak{c} \equiv \mathfrak{c} \circ \mathsf{H} \,/\, \mathsf{K}$.

8.1.3 Relation with known definitions

In this section, we examine the relation between IND-PRE-secure Functional Encryption with standard notions of security, such as indistinguishability based security. See Appendix D for a review of standard definitions of security. To begin, we show that Δ_{det}-IND-PRE-secure is equivalent to indistinguishability secure FE.

Lemma 3. *A Δ_{det}-IND-PRE-secure scheme for $\mathbf{\Sigma}_{\text{FE}}$ exists if and only if there exists an indistinguishability secure FE scheme.*

We provide a proof in Appendix D.2. Next, we show a strict separation between IND-PRE and Δ_{det}-IND-PRE security for FE. Towards this end, we exhibit an FE scheme which is indistinguishability secure but not IND-PRE secure. Combined with Lemma 3, this gives the required separation.

The idea of the separation follows that in [21]. Let $\beta : \{0,1\}^n \rightarrow \{0,1\}^n$ be a one-way permutation, and h be its hard-core predicate. Consider a function family which has only one function f. For all $x \in \{0,1\}^n$, define $f(x) := \beta(x)$. Consider an FE scheme $\mathcal{FE}$ where $\mathsf{Encrypt}(x)$ is simply a public-key encryption (PKE) of x, and the secret key for f is the secret key of the PKE scheme. ($\mathsf{Decrypt}$ first runs the decryption algorithm of PKE to obtain x, and then outputs $\beta(x)$.) In the indistinguishability game, if the adversary doesn't ask for any key, then clearly he cannot distinguish. On the other hand, if he does request a key for f, he can only send identical messages to the challenger (β is a permutation), and therefore has no advantage. Hence, $\mathcal{FE}$ is secure under the standard indistinguishability based security definition.

On the other hand, if we transform $\mathcal{FE}$ to a scheme $(\mathcal{O}, \mathcal{E})$ in the schemata framework, we show that the latter is not secure. Consider a Test algorithm which on input a bit b, chooses an n-bit string x uniformly at random, uploads message agent x and sends $b \oplus h(x)$ to the User. It also uploads a function agent corresponding to f. Thus, the ideal user sees $\beta(x)$ and $b \oplus h(x)$. Clearly, in the ideal world a PPT adversary cannot distinguish between $\mathsf{Test}(0)$ and $\mathsf{Test}(1)$, since doing so would imply guessing the hard-core bit. However, in the real world distinguishing between $\mathsf{Test}(0)$ and $\mathsf{Test}(1)$ is trivial because decryption reveals x.

Finally, one can see that if $\mathcal{O}_{\mathsf{user}}$ simply outputs $\beta(x)$ on input x, then we get a secure IND-PRE scheme.

8.2 Function-Hiding Functional Encryption

Now we turn our attention to *function-hiding* FE (with public-keys). This a significantly more challenging problem, both in terms of construction and even in terms of definition [19, 20, 1]. The difficulty in definition stems from the public-key nature of the encryption which allows the adversary to evaluate the function encoded in a key on arbitrary inputs of its choice: hence a security definition cannot insist on indistinguishability between two arbitrary functions. In prior work, this is often handled by restricting the security definition to involve functions that are chosen from a restricted class of distributions, such that the adversary's queries cannot reveal anything about the functions so chosen. The definition arising from our framework naturally generalizes this, as the security requirement applies to all hiding tests and thereby removes the need of specifying *ad hoc* restrictions. We only need to specify a schema for function-hiding FE, and the rest of the security definition follows from the framework.

The definition of the schema corresponding to function-hiding FE, $\mathbf{\Sigma}_{\text{FH-FE}} = (\mathcal{P}^{\text{FH-FE}}_{\mathsf{auth}}, \mathcal{P}^{\text{FH-FE}}_{\mathsf{user}})$, is identical to that of $\mathbf{\Sigma}_{\text{FE}}$, except that a function agent $P_C \in \mathcal{P}^{\text{FH-FE}}_{\mathsf{auth}}$ does not take any input, but always reads an input x from its communication tape and outputs $C(x)$. That is, the function agents do not reveal the function now.

We present two results – an IND-PRE-secure scheme for the class of inner-product predicates, and a Δ-IND-PRE-secure scheme for all function families. The former relies on the assumption that an IND-PRE-secure scheme for $\mathbf{\Sigma}_{\text{BGG}}$ exists, and the latter on the assumption that a Δ-secure scheme for $\mathbf{\Sigma}_{\text{OBF}}$ exists.

8.2.1 Reduction to the Generic Group Schema

The first construction is in fact an information-theoretic reduction to $\mathbf{\Sigma}_{\text{BGG}}$. Our construction is essentially the same as a construction in the recent work of [1]. This construction implemented an inner product predicate encryption scheme $\mathbb{S}_{\mathsf{IP}} = (\mathbb{S}.\mathsf{Setup}, \mathbb{S}.\mathsf{KeyGen}, \mathbb{S}.\mathsf{Encrypt}, \mathbb{S}.\mathsf{Decrypt})$ which is adaptively SIM-secure for both data and function hiding, in the generic group model. Let $\mathbb{S}.\mathsf{sim}$ denote the simulator constructed in the proof of $\mathbb{S}_{\mathsf{IP}}$. Intuitively, the simulation based proof in [1] may be interpreted as a simulation based reduction from $\mathbf{\Sigma}_{\text{FE}(\mathsf{IP})}$ to $\mathbf{\Sigma}_{\text{GG}}$ satisfying Definition 7. More details follow.

To define reduction, i.e., a scheme $(\mathcal{O}, \mathcal{E})^{\mathbf{\Sigma}_{\text{BGG}}}$, we need to translate the construction in [1] to fit the interface of $\mathcal{B}[\mathbf{\Sigma}_{\text{BGG}}]$. Note that unlike in the generic group model, $\mathcal{B}[\mathbf{\Sigma}_{\text{BGG}}]$ does not send any handles to the scheme's $\mathcal{O}$ algorithm. Instead, $\mathcal{O}$ will work with a concrete group. Let $\widehat{\mathbb{S}}$ denote the construction $\mathbb{S}$, but instantiated with the concrete group $\mathbb{Z}_q$, where q is the order of the (source and target) groups provided by $\mathbf{\Sigma}_{\text{BGG}}$.[12] Then, we define $\mathcal{O}$ as follows:

- **Encoding scheme $\mathcal{O}$:**
  - $\mathcal{O}_{\mathsf{setup}}$: Run $\widehat{\mathbb{S}}.\mathsf{Setup}$ and obtain $(\mathsf{MPK}, \mathsf{MSK})$, each of which is a vector of elements in $\mathbb{Z}_q$. Create an agent for each group element in MPK, and send it to $\mathcal{B}[\mathbf{\Sigma}_{\text{BGG}}]$ (which will send a handle for it to the user). (If there were to be entries in MPK which are not group elements, $\mathcal{O}$ sends them directly to the user.)
  - $\mathcal{O}_{\mathsf{auth}}$: Given an agent $P_f \in \mathcal{P}^{\text{FH-FE}}_{\mathsf{auth}}$, extract f from P_f and let $\mathsf{SK}_f = \widehat{\mathbb{S}}.\mathsf{KeyGen}(\mathsf{MSK}, f)$. For each group element in SK_f, send it to $\mathcal{B}[\mathbf{\Sigma}_{\text{BGG}}]$.
  - $\mathcal{O}_{\mathsf{user}}$: Given an agent $P_m \in \mathcal{P}^{\text{FH-FE}}_{\mathsf{user}}$, let $\mathsf{CT}_m = \widehat{\mathbb{S}}.\mathsf{Encrypt}(\mathsf{MPK}, m)$. Again, or each group element in CT_m, send it to $\mathcal{B}[\mathbf{\Sigma}_{\text{BGG}}]$.
- **Executer $\mathcal{E}$:** Given handles corresponding to a function agent SK_f and handles corresponding to a

[12] Groups of different orders can also be handled, but for simplicity, we consider the source and target groups to be of the same order.

message agent CT_m, $\mathcal{E}$ invokes $\mathbb{S}.\mathsf{Decrypt}$ with these handles. During the execution, $\mathbb{S}$ will require access to the generic group operations, and at the end will output a group element which is either the identity (in which case the predicate evaluates to true) or not (in which case it evaluates to false).[13] $\mathcal{E}$ will use access to $\mathcal{B}[\mathbf{\Sigma}_{\text{BGG}}]$ to carry out the group operations, and at the end carry out an equality check to find out whether the final handle output by $\mathbb{S}.\mathsf{Decrypt}$ encodes the identity or not.

Correctness follows from correctness of $\mathbb{S}$. To define our simulator, we use the simulator $\mathbb{S}.\mathsf{sim}$, which simulates the generic group oracle to a user. Our simulator is slightly simpler compared to that for $\mathbb{S}$: there, the simulator proactively checked if a group element for which a handle is to be simulated would be equal to a group element for which a handle was previously issued, and if so, used the handle again. This is because, in the generic group model, a single group element has only one representation. In our case, the simulator will issue serial numbers as handles (as $\mathcal{B}[\mathbf{\Sigma}_{\text{BGG}}]$ would have done), and equality checks are carried out (using information gathered from $\mathcal{B}[\mathbf{\Sigma}_{\text{FE}}]$) only to correctly respond to equality check requests made by the user to $\mathcal{B}[\mathbf{\Sigma}_{\text{BGG}}]$. In all other respects, our simulator is the same as the simulator in the generic group model. The proof that the simulation is good also follows the same argument as there.

8.2.2 A Construction from Obfuscation

Suppose there exists a scheme $\Pi^* = (\mathcal{O}^*, \mathcal{E}^*)$ that is a Δ-IND-PRE-secure scheme for $\mathbf{\Sigma}_{\text{OBF}}$.

- **Encoding scheme $\mathcal{O}$:**
  - $\mathcal{O}_{\mathsf{setup}}$: Generate $(\mathsf{VK}, \mathsf{SK})$ as the verification key and signing key for a signature scheme. Output VK as MPK.
  - $\mathcal{O}_{\mathsf{auth}}$: Given an agent $P_f \in \mathcal{P}^{\text{FH-FE}}_{\mathsf{auth}}$, define an agent $P'_f \in \mathcal{P}^{\text{OBF}}_{\mathsf{user}}$, which on input x outputs $f(x)$. Let $c := (\mathcal{O}^*(P'_f))$ and $\sigma = \mathsf{Sign}_{\mathsf{SK}}(c)$. Send (c, σ) to User.
  - $\mathcal{O}_{\mathsf{user}}$: Given an agent $P_m \in \mathcal{P}^{\text{FH-FE}}_{\mathsf{user}}$, define an agent $P''_{m,\mathsf{VK}} \in \mathcal{P}^{\text{OBF}}_{\mathsf{user}}$ as follows: on input (c, σ), check if $\mathsf{Verify}_{\mathsf{VK}}(c, \sigma)$ holds, and halt otherwise; if the signature does verify, invoke $\mathcal{E}^*$ with handle c and input m, and output whatever $\mathcal{E}^*$ outputs. Let $d = \mathcal{O}^*(P''_{m,\mathsf{VK}})$). Send d to User.
- **Executer $\mathcal{E}$:** Given handle (c, σ) corresponding to a function agent and a handle d corresponding to a message agent, $\mathcal{E}$ invokes $\mathcal{E}^*$ with handle d and input (c, σ). It outputs what $\mathcal{E}^*$ outputs.

The correctness of this construction is straightforward. To argue security, consider any test $\mathsf{Test} = \mathsf{D} \circ \mathfrak{c} \circ \mathfrak{s} \in \Delta$, such that Test is hiding w.r.t. $\mathbf{\Sigma}_{\text{FH-FE}}$. Then, for any PPT adversary Adv, we need to show that $\text{REAL}\langle \mathsf{Test}(0) \mid \mathcal{O} \mid \mathsf{Adv}\rangle \approx \text{REAL}\langle \mathsf{Test}(1) \mid \mathcal{O} \mid \mathsf{Adv}\rangle$. For this we consider an intermediate hybrid variable, defined as follows. Let $\widetilde{\mathfrak{s}}(0,1)$ indicate a modified version of $\mathfrak{s}$, which when given two agents P_{m_0}, P_{m_1} in $\mathcal{P}^{\text{FH-FE}}_{\mathsf{user}}$, selects P_{m_0}, but when given two agents P_{f_0}, P_{f_1} in $\mathcal{P}^{\text{FH-FE}}_{\mathsf{auth}}$, selects P_{f_1}. Then we claim that $\text{REAL}\langle \mathsf{Test0} \mid \mathcal{O} \mid \mathsf{Adv}\rangle \approx \text{REAL}\langle \mathsf{D} \circ \mathfrak{c} \circ \widetilde{\mathfrak{s}} \mid \mathcal{O} \mid \mathsf{Adv}\rangle \approx \text{REAL}\langle \mathsf{Test1} \mid \mathcal{O} \mid \mathsf{Adv}\rangle$. For simplicity, consider D which only outputs a single pair of function agents (P_{f_0}, P_{f_1}) and a single pair of message agents (P_{m_0}, P_{m_1}). (The general case is handled using a sequence of hybrids, in a standard way.)

To show the first approximate equality, consider a test Test' and adversary Adv' which work as follows. Test' internally simulates $\mathsf{Test}(0)$ and $\mathcal{O}$ with the following differences: when $\mathcal{O}_{\mathsf{setup}}$ outputs the signing key SK, Test' forwards it to Adv'; when the two agents P_{f_0}, P_{f_1} are sent to $\mathfrak{s}$, $\mathsf{Test}'(b)$ outputs P_{f_b} to $\mathcal{B}[\mathbf{\Sigma}_{\text{OBF}}]$ (or $\mathcal{O}^*$). Adv', when it receives c from $\mathcal{O}^*$, first signs it using SK to obtain σ, and then passes

[13] Though not the case with the construction in [1], a general algorithm in the generic group model may check for identities by comparing handles, not just at the end, but at any point during its execution. In this case, $\mathcal{E}$ should proactively check every handle it receives from $\mathcal{B}[\mathbf{\Sigma}_{\text{BGG}}]$ against all previously received handles, to see if they encode the same group element; if so, the newly received handle is replaced with the existing one.

on (c, σ) to an internal copy of Adv; otherwise, it lets Adv directly interact with Test'. It can be seen that $\text{REAL}\langle\mathsf{Test}'(0) \mid \mathcal{O}^* \mid \mathsf{Adv}'\rangle = \text{REAL}\langle\mathsf{Test}(0) \mid \mathcal{O} \mid \mathsf{Adv}\rangle$ and $\text{REAL}\langle\mathsf{Test}'(1) \mid \mathcal{O}^* \mid \mathsf{Adv}'\rangle = \text{REAL}\langle\mathsf{D} \circ \mathfrak{c} \circ \widetilde{\mathfrak{s}} \mid \mathcal{O} \mid \mathsf{Adv}\rangle$. Further, $\mathsf{Test}' \in \Delta$. Also, it is easy to see that if Test' is not hiding w.r.t. $\mathbf{\Sigma}_{\text{OBF}}$, then Test is not hiding w.r.t. $\mathbf{\Sigma}_{\text{FH-FE}}$ (because User's interface to $\mathbf{\Sigma}_{\text{FH-FE}}$ can be used to emulate its interface to $\mathbf{\Sigma}_{\text{OBF}}$). Thus, if Test is hiding w.r.t. $\mathbf{\Sigma}_{\text{FH-FE}}$, then $\text{REAL}\langle\mathsf{Test}'(0) \mid \mathcal{O}^* \mid \mathsf{Adv}'\rangle \approx \text{REAL}\langle\mathsf{Test}'(1) \mid \mathcal{O}^* \mid \mathsf{Adv}'\rangle$. This establishes that $\text{REAL}\langle\mathsf{Test0} \mid \mathcal{O} \mid \mathsf{Adv}\rangle \approx \text{REAL}\langle\mathsf{D} \circ \mathfrak{c} \circ \widetilde{\mathfrak{s}} \mid \mathcal{O} \mid \mathsf{Adv}\rangle$.

To show that $\text{REAL}\langle\mathsf{D} \circ \mathfrak{c} \circ \widetilde{\mathfrak{s}} \mid \mathcal{O} \mid \mathsf{Adv}\rangle \approx \text{REAL}\langle\mathsf{Test1} \mid \mathcal{O} \mid \mathsf{Adv}\rangle$, we consider another test Test''. Now, Test'' internally simulates $\mathsf{Test}(1)$ and $\mathcal{O}$ with the following differences: when the two message agents P_{m_0}, P_{m_1} are sent to $\mathfrak{s}$, $\mathsf{Test}''(b)$ sends $(P''_{m_0,\mathsf{VK}}, P''_{m_1,\mathsf{VK}})$ to Adv'' and outputs $P''_{m_b,\mathsf{VK}}$ to $\mathcal{B}[\mathbf{\Sigma}_{\text{OBF}}]$ (or $\mathcal{O}^*$), where $P''_{m_b,\mathsf{VK}}$ was as defined in the description of $\mathcal{O}_{\mathsf{user}}$. Then, $\text{REAL}\langle\mathsf{Test}''(0) \mid \mathcal{O}^* \mid \mathsf{Adv}\rangle = \text{REAL}\langle\mathsf{D} \circ \mathfrak{c} \circ \widetilde{\mathfrak{s}} \mid \mathcal{O} \mid \mathsf{Adv}\rangle$ and $\text{REAL}\langle\mathsf{Test}''(1) \mid \mathcal{O}^* \mid \mathsf{Adv}\rangle = \text{REAL}\langle\mathsf{Test}(1) \mid \mathcal{O} \mid \mathsf{Adv}\rangle$. Also, as before, $\mathsf{Test}'' \in \Delta$. If Test'' is hiding w.r.t. $\mathbf{\Sigma}_{\text{OBF}}$, then we can conclude that $\text{REAL}\langle\mathsf{Test}''(0) \mid \mathcal{O}^* \mid \mathsf{Adv}\rangle \approx \text{REAL}\langle\mathsf{Test}''(1) \mid \mathcal{O}^* \mid \mathsf{Adv}\rangle$, and hence $\text{REAL}\langle\mathsf{D} \circ \mathfrak{c} \circ \widetilde{\mathfrak{s}} \mid \mathcal{O} \mid \mathsf{Adv}\rangle \approx \text{REAL}\langle\mathsf{Test1} \mid \mathcal{O} \mid \mathsf{Adv}\rangle$. Thus it only remains to show that Test'' is hiding w.r.t. $\mathbf{\Sigma}_{\text{OBF}}$. Firstly, by the security of the signature scheme, for any PPT adversary User, w.h.p., it does not query $\mathbf{\Sigma}_{\text{OBF}}$ with a handle and an input (c, σ) for a c that was not produced by Test''. Now, conditioned on this event, if User distinguishes $\mathsf{Test}''(0)$ and $\mathsf{Test}''(1)$, this User can be turned into one that distinguishes between $\mathsf{Test}(0)$ and $\mathsf{Test}(1)$ when interacting with $\mathbf{\Sigma}_{\text{FH-FE}}$. Thus, since Test is hiding w.r.t. $\mathbf{\Sigma}_{\text{FH-FE}}$, it follows that Test'' is hiding w.r.t. $\mathbf{\Sigma}_{\text{OBF}}$, as was required to be shown.

9 Relating Cryptographic Agents to other Cryptographic Objects

Various interesting cryptographic objects such as fully homomorphic encryption, property preserving encryption, private key functional encryption, witness encryption, the many variants of functional encryption and such others are readily obtained from $\mathsf{IND\text{-}PRE}$ secure implementations of corresponding schemata. In this section, we provide two examples – fully homomorphic encryption and property preserving encryption. The details of other examples are left to the full version.

9.1 Fully Homomorphic Encryption

In this section, we construct a cryptographic agent schema $\mathbf{\Sigma}_{\text{FHE}}$ for Fully Homomorphic Encryption (FHE). For a message space $\mathcal{X} = \{\mathcal{X}\}_\kappa$ and a circuit family $\mathcal{F} = \{\mathcal{F}\}_\kappa$, we define the schema $\mathbb{P}_{\text{FHE}} = (\mathcal{P}^{\text{FHE}}_{\mathsf{test}}, \mathcal{P}^{\text{FHE}}_{\mathsf{user}})$ as follows:

- An agent $P^{\mathsf{Msg}} \in \mathcal{P}^{\text{FHE}}_{\mathsf{user}}$ is specified as follows: It's initial message x is put in its parameter tape. When invoked with an input C on its input tape, it reads a set of messages $x_2, x_3, \ldots, x_t$ from its communication tapes. Then it computes $C(x_1, .., x_t)$ where x_1 is its own message (either from the work-tape, or if the work-tape is empty, from its parameter tape). Then it updates its work-tape with this value. When invoked without an input, it sends its message to the first program in the session.
- An agent $P^{\mathsf{Dec}} \in \mathcal{P}^{\text{FHE}}_{\mathsf{auth}}$ is defined as follows: when executed with an agent P^{Msg} it reads from its communication tape a single message from P^{Msg} and outputs it[14].

Next, we claim that a semantically secure FHE scheme $\mathbb{S}_{\text{FHE}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Eval})$ can be naturally constructed from a $\Delta_{\mathsf{det}}\text{-}\mathsf{IND\text{-}PRE}$ secure scheme for $\mathbf{\Sigma}_{\text{FHE}}$. We provide the construction in Appendix E.

[14]Note that there is no parameter to a $\mathcal{P}_{\mathsf{auth}}$ agent as there is only one of its kind. However, we can allow a single schema to capture multiple FHE schemes with independent keys, in which case an index for the key would be the parameter for $\mathcal{P}_{\mathsf{auth}}$ agents.

9.2 Property Preserving Encryption

In this section, we present a cryptographic agent schema $\mathbf{\Sigma}_{\mathsf{PPE}}$ for property preserving encryption(PPE). Definitions for PPE algorithms and security can be found in Appendix F.

Let $P : \mathcal{M} \times \mathcal{M} \to \{0,1\}$ be a (polynomial-time computable) binary property over the message space $\mathcal{M}$. The schema $\mathbf{\Sigma}_{\mathsf{PPE}} = (\mathcal{P}^{\mathsf{PPE}}_{\mathsf{auth}}, \emptyset)$, where $\mathcal{P}^{\mathsf{PPE}}_{\mathsf{auth}}$ has two kinds of agents, denoted by P^{Msg} and P^{Dec}, is defined as follows:

- P^{Msg} for a message $m \in \mathcal{M}$ is specified as follows: it has a message m on its parameter tape. When invoked with a command $\mathsf{compute}$ on its input tape, it reads a message m' from its communication tape, computes $P(m, m')$, outputs it and halts. When invoked with a command send, it sends its message m to the first agent in the session.
- P^{Dec} reads from its communication tape a single message and outputs it.

In Appendix F we show that Δ_{det}-IND-PRE and LoR security notions are equivalent for PPE.

10 On Bypassing Impossibilities

An important aspect of our framework is that it provides a clean mechanism to tune the level of security for each primitive to a "sweet spot." The goal of such a definition is that it should imply prevalent achievable definitions while bypassing known impossibilities. The tuning is done by defining the family of tests, Γ with respect to which IND-PRE security is required. Below we discuss a few schemata and the definitions we recommend for them, based on what is known to be impossible.

Obfuscation. As we show in Section 7, an IND-PRE-secure scheme for $\mathbf{\Sigma}_{\mathrm{OBF}}$ cannot exist. The impossibility proof relies on the fact that the test can upload an agent with (long) secrets in them. However, this argument stops applying when we restrict ourselves to tests in Δ: a test in Δ has the structure $\mathsf{D} \circ \mathfrak{c} \circ \mathfrak{s}$ and $\mathfrak{c}$ will reveal the agent to User. Note that then there could be at most one bit of uncertainty as to which agent was uploaded.

We point out that while Δ-IND-PRE-security may now appear restrictive (compared to the impossibly strong IND-PRE-security), it is still much stronger than the prevalent notions of indistinguishability obfuscation and differing inputs obfuscation, introduced by Barak et al. [9]. Indeed, to the best of our knowledge, it yields the first definition of general obfuscation, since the seminal work of Barak et al., that can plausibly exist for all functions.

We also observe that Δ-IND-PRE-secure obfuscation (or equivalently, adaptive differing-inputs obfuscation) is easier to use in constructions than differing-inputs obfuscation, as exemplified by our constructions in Section 8.1.2 and Section 8.2.2.

Functional Encryption. Public-key function-hiding FE, as modeled by $\mathbf{\Sigma}_{\mathrm{FH\text{-}FE}}$, is a stronger primitive than obfuscation (for the same class of functions), as the latter can be easily reduced to the former. This means that there is no IND-PRE-secure scheme for $\mathbf{\Sigma}_{\mathrm{FH\text{-}FE}}$ for general functions. We again consider Δ-IND-PRE security as a sweet-spot for defining function-hiding functional encryption. Indeed, prior to this definition, arguably there was no satisfactory definition for this primitive. Standard indistinguishability based definitional approaches (which typically specify an explicit test that is ideal-hiding) runs into the problem that if the user is allowed to evaluate a given function on any inputs of its choice, there is no one natural ideal-hiding test. Prior works have proposed different approaches to this problem: by restricting to only a specific test [19, 20], or using a relaxed simulation-based definition [1, 42]. Δ-IND-PRE security implies the definitions of Boneh et al. [19, 20], but is in general incomparable with the simulation-based definitions in the other works. These latter definitions can be seen as using a test in the ideal world that is a more generous version of the test in the real world, thereby allowing an adversary to learn more information than would in general be considered ideal. Our definition does not suffer from such information leakage.

For non-function-hiding FE (captured by the schema $\mathbf{\Sigma}_{\text{FE}}$) too, there are many known impossibility results, when simulation-based security definitions are used [21, 13, 5]. At a high-level, these impossibilities followed a "compression" argument – the decryption of the challenge CT with the queried keys comprise a pseudorandom string R, but the adversary's key queries and challenge message are sequenced in such a way that to simulate its view, the simulator must somehow compress R significantly. These arguments do not apply to IND-PRE-security simply for the reason that there is no simulator implied by it. We do not have any candidate constructions for IND-PRE-secure scheme for $\mathbf{\Sigma}_{\text{FE}}$, for general functions, but we leave open the possibility that it exists. We do however, provide a construction for a Δ-IND-PRE-secure scheme for $\mathbf{\Sigma}_{\text{FE}}$, assuming one for $\mathbf{\Sigma}_{\text{OBF}}$ (by means of a Δ-reduction).

Generic Group and Random Oracle. It is well known that a proof of security in the generic group or the random oracle model provides only a heuristic security guarantee. Several works have shown that these oracles are "uninstantiable," and further there are uninstantiable primitives that can be implemented in the models with such oracles [38, 48, 70, 46, 11]. These results do not contradict Assumption 1, however, because the primitives in question, like non-commiting encryptions, zero-knowledge proofs and even signature schemes, do not fit into our framework of schemata. In other words, despite its generality, schemata can be used to model only certain kind of primitives, which seem insufficient to imply such separations between the generic group model and the standard model. As such, we propose Assumption 1, with $\Gamma = \Gamma_{\mathsf{ppt}}$, the family of all PPT tests, as an assumption worthy of investigation. However, the weaker assumption, with $\Gamma = \Delta$ suffices for our construction in Section 8.2.1, if we settle for Δ-IND-PRE security for the resulting scheme.

11 Conclusions and Open Problems

In this work, we provided a general unifying framework to model various cryptographic primitives and their security notions, along with powerful reduction and composition theorems. Our framework easily captures seemingly disparate objects such as obfuscation, functional encryption, fully homomorphic encryption, property preserving encryption as well as idealized models such as the generic group model and the random oracle model.

Given that various cryptographic primitives can all be treated as objects of the same kind (schema), it is natural to compare them with each other. We have shown that obfuscation is complete (under standard computational assumptions), but completely leave open the question of *characterizing* complete schemata. We also raise the question of characterizing *trivial* schemata — those which can be reduced to the null schema — as well as characterizing *realizable* schemata — those which have (say) IND-PRE-secure schemes.

We presented a hierarchy of security notions $\{\Delta^*\text{-IND-PRE}, \Delta_{\mathsf{det}}\text{-IND-PRE}\} \leq \Delta\text{-IND-PRE} \leq \text{IND-PRE} \leq \text{SIM}$ defined using various test families (or, in the case of SIM, as a reduction to the null-schema), but the relationships between these for any given schema are not fully understood. We leave it as an open problem to provide separations between these various notions of security for various schemata. For the case of functional encryption we provide a separation of Δ_{det}-IND-PRE from IND-PRE. For obfuscation we conjecture that all the above notions are different from each other for some function family.

Finally, while we provide several instantiations of our framework, there are several primitives that our framework does not capture as-is, such as signatures, CCA secure encryption, obfuscation with security against malicious obfuscators, non-committing encryption and such others. It is an important open problem to extend our framework to support modeling the above primitives.

References

[1] S. Agrawal, S. Agrawal, S. Badrinarayanan, A. Kumarasubramanian, M. Prabhakaran, and A. Sahai. Function private functional encryption and property preserving encryption : New definitions and positive results. Cryptology Eprint Arxiv, 2013. 6, 19, 22, 23, 25, 31, 32

[2] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010. 32

[3] S. Agrawal, D. Boneh, and X. Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *CRYPTO*, pages 98–115, 2010. 32

[4] S. Agrawal, D. M. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Asiacrypt*, 2011. 32

[5] S. Agrawal, S. Gurbanov, V. Vaikuntanathan, and H. Wee. Functional encryption: New perspectives and lower bounds. In *Crypto*, 2013. 6, 26

[6] J. Alwen, M. Barbosa, P. Farshim, R. Gennaro, S. D. Gordon, S. Tessaro, and D. A. Wilson. On the relationship between functional encryption, obfuscation, and fully homomorphic encryption. In *IMA Int. Conf.*, pages 65–84, 2013. 32

[7] P. Ananth, D. Boneh, S. Garg, A. Sahai, and M. Zhandry. Differing-inputs obfuscation and applications. Cryptology Eprint Arxiv, 2013. http://eprint.iacr.org/2013/689.pdf. 6, 17, 19, 20, 31, 32

[8] B. Barak, S. Garg, Y. T. Kalai, O. Paneth, and A. Sahai. Protecting obfuscation against algebraic attacks. In *Eurocrypt*, 2014. 16

[9] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001. 5, 16, 17, 25, 31

[10] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, May 2012. 17

[11] M. Bellare, A. Boldyreva, and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *EUROCRYPT*, pages 171–188, 2004. 26

[12] M. Bellare, V. T. Hoang, and S. Keelveedhi. Instantiating random oracles via uces. In *Crypto*, 2013. 15

[13] M. Bellare and A. O'Neill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In *CANS*, pages 218–234, 2013. 6, 26

[14] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the First Annual Conference on Computer and Communications Security*. ACM, November 1993. 15

[15] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007. 32

[16] N. Bitansky, R. Canetti, O. Paneth, and A. Rosen. Indistinguishability obfuscation vs. auxiliary-input extractable functions: One must fall. Cryptology ePrint Archive, Report 2013/641, 2013. http://eprint.iacr.org/. 32

[17] A. Boldyreva, D. Cash, M. Fischlin, and B. Warinschi. Foundations of non-malleable hash and one-way functions. In *In ASIACRYPT*, pages 524–541, 2009. 15

[18] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001. 32

[19] D. Boneh, A. Raghunathan, and G. Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In *CRYPTO*, 2013. 6, 22, 25

[20] D. Boneh, A. Raghunathan, and G. Segev. Function-private subspace-membership encryption and its applications. In *Asiacrypt*, 2013. 6, 22, 25

[21] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011. 6, 21, 26, 38

[22] D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. *IACR Cryptology ePrint Archive*, 2002:80, 2002. 15

[23] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007. 32

[24] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, pages 290–307, 2006. 32

[25] E. Boyle, K.-M. Chung, and R. Pass. On extractability obfuscation. In *TCC*, 2014. http://eprint.iacr.org/2013/650.pdf. 6, 19, 20, 31

[26] E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In *Public Key Cryptography*, pages 501–519, 2014. 19

[27] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *CRYPTO*, 2012. 31, 32

[28] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012. 31, 32

[29] Z. Brakerski and G. N. Rothblum. Obfuscating conjunctions. In *CRYPTO*, pages 416–434, 2013. 31

[30] Z. Brakerski and G. N. Rothblum. Black-box obfuscation for d-cnfs. In *ITCS*, 2014. http://eprint.iacr.org/. 31

[31] Z. Brakerski and G. N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *TCC*, 2014. http://eprint.iacr.org/. 16, 31

[32] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, pages 501–521, 2011. 31, 32

[33] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011. 31, 32

[34] Z. Brakerski and V. Vaikuntanathan. Lattice-based FHE as secure as PKE. In *ITCS*, 2014. http://eprint.iacr.org/2013/541. 31, 32

[35] R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *CRYPTO*. Springer Berlin Heidelberg, 1997. 15, 31

[36] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, 2001. 4

[37] R. Canetti and R. R. Dakdouk. Obfuscating point functions with multibit output. In *EUROCRYPT*, 2008. 15, 31

[38] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *STOC*, pages 209–218, 1998. 26

[39] R. Canetti, D. Micciancio, and O. Reingold. Perfectly one-way probabilistic hash functions (preliminary version). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, 1998. 15, 31

[40] R. Canetti, G. Rothblum, and M. Varia. Obfuscation of hyperplane membership. In *TCC*, 2010. 31

[41] R. Canetti and V. Vaikuntanathan. Obfuscating branching programs using black-box pseudo-free groups. Cryptology ePrint Archive, Report 2013/500, 2013. http://eprint.iacr.org/. 31

[42] A. D. Caro and V. Iovino. On the power of rewinding simulators in functional encryption. Cryptology ePrint Archive, Report 2013/752, 2013. http://eprint.iacr.org/. 25

[43] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010. 32

[44] S. Chatterjee and M. P. L. Das. Property preserving symmetric encryption: Revisited. *IACR Cryptology ePrint Archive*, 2013:830, 2013. http://eprint.iacr.org/2013/830. 32

[45] C. Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001. 32

[46] A. W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. *IACR Cryptology ePrint Archive*, 2002:86, 2002. 26

[47] M. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43, 2010. Full Version in http://eprint.iacr.org/2009/616.pdf. 31, 32

[48] M. Fischlin. A note on security proofs in the generic model. In T. Okamoto, editor, *ASIACRYPT*, Lecture Notes in Computer Science, pages 458–469. Springer, 2000. 26

[49] S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, 2013. 15, 31

[50] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013. http://eprint.iacr.org/. 17, 19, 20, 31, 32

[51] S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO*, 2013. 32

[52] S. Garg, C. Gentry, S. Halevi, and D. Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. Cryptology Eprint Arxiv, 2014. http://eprint.iacr.org/2013/860.pdf. 32

[53] C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. 40

[54] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009. 31, 32

[55] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008. 32

[56] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013. 31, 32

[57] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In ACM, editor, *STOC*, pages 218–229, 1987. 4

[58] S. Goldwasser, V. Goyal, A. Jain, and A. Sahai. Multi-input functional encryption. https://eprint.iacr.org/2013/727.pdf. 17, 19

[59] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. How to run turing machines on encrypted data. In *CRYPTO (2)*, pages 536–553, 2013. 32

[60] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, pages 555–564, 2013. 32

[61] S. Goldwasser and G. N. Rothblum. On best-possible obfuscation. In *Proceedings of the 4th Conference on Theory of Cryptography*, TCC'07, 2007. 31

[62] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute based encryption for circuits. In *STOC*, 2013. 32

[63] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006. 32

[64] D. Hofheinz, J. Malone-lee, and M. Stam. Obfuscation for cryptographic purposes. In *In TCC*, pages 214–232, 2007. 31

[65] S. Hohenberger, G. N. Rothblum, A. Shelat, and V. Vaikuntanathan. Securely obfuscating re-encryption. In *Proceedings of the 4th Conference on Theory of Cryptography*, TCC'07, 2007. 31, 32

[66] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008. 32

[67] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010. 32

[68] B. Lynn, M. Prabhakaran, and A. Sahai. Positive results and techniques for obfuscation. In *EUROCRYPT*, 2004. 31

[69] U. Maurer. Abstract models of computation in cryptography. In *IMA Int. Conf.*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005. 15

[70] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO*, pages 111–126, 2002. 26

[71] A. O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/. 38

[72] O. Pandey and Y. Rouselakis. Property preserving symmetric encryption. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology âĂŞ EUROCRYPT 2012*, number 7237 in Lecture Notes in Computer Science, pages 375–391. Springer Berlin Heidelberg, Jan. 2012. 31, 32, 41

[73] A. Sahai and B. Waters. Functional encryption:beyond public key cryptography. Power Point Presentation, 2008. http://userweb.cs.utexas.edu/ËIJbwaters/presentations/ files/functional.ppt. 31

[74] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005. 31

[75] A. Sahai and B. Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. In *Crypto*, 2013. http://eprint.iacr.org/2013/454.pdf. 31

[76] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984. 32

[77] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Eurocrypt*, pages 256–266. Springer-Verlag, 1997. 15

[78] B. Waters. Functional encryption for regular languages. In *Crypto*, 2012. 32

[79] H. Wee. On obfuscating point functions. In *STOC*, pages 523–532, 2005. 15, 31

A Related Work

Recent times have seen a fantastic boom in the area of *computing with encrypted information*. Several exciting primitives supporting advanced functionalities, such as fully homomorphic encryption [54, 47, 32, 33, 28, 27, 34, 56], functional encryption [74, 73], property preserving encryption [72, 1] have been constructed. Some functionalities require the data to be hidden but permit the function to be public, while others, most notably program obfuscation [9], permit the data to be public but the function to be hidden. Here, we review the state of the art in these fields.

Program Obfuscation. Program Obfuscation is the task of garbling a given program so that the input-output behavior is retained, but everything else about the program is hidden. The formal study of program obfuscation was initiated by Barak et al. [9] who showed that the strongest possible notion of security, called *virtual black box* security was impossible to achieve for general circuits. To address this, they defined weaker notions of security, such as *indistinguishability obfuscation* (denoted by I-Obf), which states that for two equivalent circuits C_0 and C_1, their obfuscations should be computationally indistinguishable. A related but stronger security notion defined by [9] was that of *differing input obfuscation* (denoted by DI-Obf), which further requires that an adversary who can distinguish between C_0 and C_1 can be used to *extract* an input on which the two circuits differ.

Despite these weakenings, the area of program obfuscation was plagued by impossibilities [68, 61, 64] for a long time, with few positive results, often for very specialized classes of functions [35, 39, 79, 40, 65, 37]. This state of affairs however, has improved significantly in recent times, when constructions of graded encoding schemes [49] were leveraged to build program obfuscators for complex functionalities, such as conjunctions [29], d-CNF formulas [30], circuits [50, 31, 41] and even Turing machines [7] in weaker models of computation such as the generic graded encoding scheme model [29, 30, 31, 7], the generic colored matrix model [50] and the idealized pseudo free group model [41].

These constructions are proven secure under different notions of security : virtual black box, I-Obf, DI-Obf. Alongside, several new applications have been developed for IP-Obf [75] and DI-Obf [7, 25]. There is a growing research effort in exploring the plausibility and connections between different notions of obfuscation [16, 52]. A better understanding of various notions of obfuscation and connections with various related notions such as functional encryption (see below), is slowly emerging, with much promise for the future.

Functional Encryption. Functional encryption generalizes public key encryption to allow fine grained access control on encrypted data. In functional encryption, a user can be provided with a secret key corresponding to a function f, denoted by SK_f. Given SK_f and ciphertext $\mathsf{CT}_x = \mathsf{Encrypt}(x)$, the user may run the decryption procedure to learn $f(x)$. Security of the system guarantees that nothing beyond $f(x)$ can be learned from CT_x and SK_f. Functional encryption systems traditionally focussed on restricted classes of functions such as the identity function [76, 18, 45, 24, 55, 43, 2, 3], membership checking [23], boolean formulas [63, 15, 67], inner product functions [66, 67, 4] and more recently, even regular languages [78]. Recent times saw constructions for more general classes of functions: Gurabov et al. [62] and Garg et al. [51] provided the first constructions for an important subclass of FE called "public index FE" for all circuits, Goldwasser et al. [60] constructed succinct simulation-secure single-key FE scheme for all circuits, Garg et al. [50] constructed multi-key FE schemes for all circuits while Goldwasser et al. and Ananth et al. [59, 7] also constructed FE for Turing machines.

Functional Encryption and Obfuscation are not just powerful cryptographic primitives in their own right, but are also intimately related objects – for example, it was shown in [50] that indistinguishability obfuscation implies functional encryption. Recently, differing input obfuscation has been used to construct FE for Turing machines [59].

Fully homomorphic encryption. Fully homomorphic encryption allows a user to evaluate a circuit C on encrypted messages $\{\mathsf{CT}_i = \mathsf{Encrypt}(x_i)\}_{i\in[n]}$ so that $\mathsf{Decrypt}(C(\mathsf{CT}_1, \ldots, \mathsf{CT}_n)) = C(x_1, \ldots, x_n)$. Since the first breakthrough construction by Gentry [54], extensive research effort has been focused on providing improvements [47, 32, 33, 28, 27, 34, 56].

Recently, Alwen et al. [6] explored the connections between FHE, FE and obfuscation. In [6], the authors introduce the notion of randomized FE which can be used to construct FHE. In addition, they explore the problem of obfuscating specific re-encryption functionalities, introducing new notions extending those proposed in earlier works on re-encryption [65]. They also develop techniques to use obfuscated re-encryption circuits to construct FE schemes.

Property Preserving Encryption. The notion of property preserving encryption (PPE) was introduced in a very recent work by Pandey and Rouselakis [72]. Property preserving encryption is a symmetric key primitive, which permits some pre-determined property $P(x_1, x_2)$ to be publicly tested given only the ciphertexts $\mathsf{CT}(x_1), \mathsf{CT}(x_2)$. In [72], the authors formalize the notion of PPE, provide definitions of security and provide a candidate construction for *inner product* PPE in the generic group model. Subsequently [44] displayed an attack against the construction in [72], which was fixed in [1]. Agrawal et al. [1] also provide the first standard model construction of PPE.

This rich body of primitives is interdependent not only in terms of philosophy and techniques, but also in terms of *non-interaction.* Unlike the case of multiparty computation, where a user (in general) continues to send and receive messages throughout the protocol, the above primitives do not permit users to "keep playing". A user may create an *obfuscated agent* once and for all, and then release it into the wild. This agent is expected to reveal nothing other than what is permitted by its functionality, but must interface in a well defined manner with other agents or expected inputs.

Another aspect to note, is that many of the above primitives are known to be impossible to instantiate under the strong *simulation based security* desired by MPC. Indeed, positive results often settle for a weaker *indistinguishability based security*, which is also the focus of this work.

B Obfuscation Schema is Complete

In this section we prove that $\mathbf{\Sigma}_{\text{OBF}}$ is "complete" under the notion of reduction defined in Definition 7. More precisely, we show two kinds of reductions.

1. Any schema $(\emptyset, \mathcal{P})$ in which the agents are *non-interactive* (but possibly randomized and reactive) has a reduction $(\mathcal{O}, \mathcal{E})$ to $\mathbf{\Sigma}_{\text{OBF}}$ in which $\mathcal{O}$ is setup-free.
2. Any schema $\mathbf{\Sigma} = (\mathcal{P}_{\mathsf{auth}}, \mathcal{P}_{\mathsf{user}})$ (possibly with $\mathcal{P}_{\mathsf{test}} \neq \mathcal{P}_{\mathsf{user}}$ and containing possibly randomized, reactive, interactive agents) has a reduction $(\mathcal{O}, \mathcal{E})$ to $\mathbf{\Sigma}_{\text{OBF}}$ in which $\mathcal{O}$ has setup.

We point out that if $\mathcal{P}_{\mathsf{test}} \neq \mathcal{P}_{\mathsf{user}}$, in general it is necessary that $\mathcal{O}$ has setup, as otherwise an adversarial user can create obfuscations of programs in $\mathcal{P}_{\mathsf{auth}}$ itself.

We sketch each of these reductions below. The security of these reductions only depend on standard symmetric-key and public-key cryptography primitives. The proofs are conceptually clean and simple, as the reductions between schemata occur in an idealized world. However, the detailed descriptions of the reductions and the simulator tend to be somewhat long. We provide some of the details to clarify subtleties and also to illustrate the nature of the reductions and proofs, but defer further details to the full version.

We carry out the reduction in two steps: first we show how to reduce any schema with non-interactive agents to $\mathbf{\Sigma}_{\text{OBF}}$, and then build on it to reduce all schemata (including those with interactive agents) to $\mathbf{\Sigma}_{\text{OBF}}$.

B.1 Construction for Non-Interactive Agents

In this section we reduce any schema of the form $\mathbf{\Sigma}_{RR} = (\emptyset, \mathcal{P})$, in which the agents are randomized and reactive, but non-interactive, to $\mathbf{\Sigma}_{\text{OBF}}$. For this we define an intermediate schema, $\mathbf{\Sigma}_R$ of randomized, but non-reactive, non-interactive agents, and give two reductions: we reduce $\mathbf{\Sigma}_{RR}$ to $\mathbf{\Sigma}_R$ and $\mathbf{\Sigma}_R$ to $\mathbf{\Sigma}_{\text{OBF}}$. These can then be composed together using the transitivity of reducibility (Theorem 3) to obtain our reduction from $\mathbf{\Sigma}_{RR}$ to $\mathbf{\Sigma}_{\text{OBF}}$.[15]

Below, we will write $\mathbf{\Sigma}_0$ for $\mathbf{\Sigma}_{\text{OBF}}$, $\mathbf{\Sigma}_1$ for $\mathbf{\Sigma}_R$, and $\mathbf{\Sigma}_2$ for $\mathbf{\Sigma}_{RR}$.

$(\mathcal{O}_1, \mathcal{E}_1)$ to reduce $\mathbf{\Sigma}_1$ to $\mathbf{\Sigma}_0$. On receiving a randomized agent P_1 from Test, $\mathcal{O}_1$ uploads the following (deterministic) agent P_0 to $\mathcal{B}[\mathbf{\Sigma}_0]$: P_0 has the parameters of P_1 as well as a freshly chosen seed s for a pseudorandom function (PRF) built-in as its parameters; when invoked it interprets its input as (i, x), generates a random tape for P_1 using the PRF applied to (i, x), as $r = \mathsf{PRF}_s(i, x)$, and executes $P_1(x; r)$. (The κ-bit index i is used to implement multiple independent executions of the randomized agent with the same input.)

$\mathcal{E}_1$ translates User's interaction with $\mathcal{B}[\mathbf{\Sigma}_1]$ to an interaction with $\mathcal{B}[\mathbf{\Sigma}_0]$: when User requests to upload a randomized agent to $\mathcal{B}[\mathbf{\Sigma}_1]$, $\mathcal{E}_1$ will upload to $\mathcal{B}[\mathbf{\Sigma}_0]$ an agent as created by $\mathcal{O}_1$. When $\mathcal{B}[\mathbf{\Sigma}_0]$ sends $\mathcal{E}_1$ a handle, it forwards it to User. When User sends an execution command with a handle h and an input x to $\mathcal{B}[\mathbf{\Sigma}_1]$, $\mathcal{E}_1$ translates it to the handle h and input (i, x) for $\mathcal{B}[\mathbf{\Sigma}_0]$, where i is a randomly chosen κ-bit index. The correctness of the reduction follows directly from the security of the PRF, and the fact that it is unlikely that $\mathcal{E}_1$ will choose the same value for i in two different sessions.

The simulator $\mathcal{S}_1$, which translates Adv's interaction with $\mathcal{B}[\mathbf{\Sigma}_0]$ to an interaction with $\mathcal{B}[\mathbf{\Sigma}_1]$, behaves as follows: it passes on handles it receives from $\mathcal{B}[\mathbf{\Sigma}_1]$ as handles from $\mathcal{B}[\mathbf{\Sigma}_0]$. If the user sends an upload command, $\mathcal{S}_1$ will upload the agent as it is (since $\mathbf{\Sigma}_1$ allows deterministic agents as well). $\mathcal{S}_1$ also maintains a list of the form (h, i, x, y) where h is a handle obtained that does not correspond to an

[15]The tape and time bounds for the agents in $\mathbf{\Sigma}_{\text{OBF}}$ will depend on the tape and time bounds for the schema $\mathbf{\Sigma}_{RR}$. For simplicity, we leave this bound to be only implicitly specified by our reductions.

agent uploaded by Adv, (i, x) is an input for h from a session execution command given by Adv, and y is the output it reported back to Adv for that session. On receiving a new session request $h(z)$, i.e., for an agent handle h with input z, $\mathcal{S}_1$ behaves differently depending on whether h is a handle that corresponds to an agent uploaded by Adv, or not. In the former case, $\mathcal{S}_1$ simply forwards the request $h(z)$ to $\mathcal{B}[\mathbf{\Sigma}_1]$ and returns the response from $\mathcal{B}[\mathbf{\Sigma}_1]$ back to Adv. In the latter case, $\mathcal{S}_1$ interprets z as (i, x); then, if there is an entry of the form (h, i, x, y) in its list, $\mathcal{S}_1$ returns y to Adv; else it forwards the session request (h, x) to $\mathbf{\Sigma}_0$, and gets back (a fresh) output y, records (h, i, x, y) in its list, and sends y to User. It is easy to show, from the security of the PRF, that $\mathcal{S}_1$ satisfies the correctness requirements.

$(\mathcal{O}_2, \mathcal{E}_2)$ to reduce $\mathbf{\Sigma}_2$ to $\mathbf{\Sigma}_1$. We omit the detailed description of $\mathcal{O}_2, \mathcal{E}_2$ and the simulator $\mathcal{S}_2$ associated with this reduction, but instead just describe the behavior of the non-reactive agent P_1 that $\mathcal{O}_2$ sends to $\mathcal{B}[\mathbf{\Sigma}_1]$, when given a reactive agent P_2 of schema $\mathbf{\Sigma}_2$.

The idea is that the reactive agent P_2 can be implemented by a non-reactive agent P_1 which outputs an encrypted configuration of P_2 that can then be fed back as input to P_1. More precisely, P_1 will contain the parameters of P_2 and keys for a semantically secure symmetric-key encryption scheme and a message authentication code (MAC) built-in as its own parameters. If invoked with just an input for P_2, P_1 considers this an invocation of P_2 from its start configuration. In this case, P_1 uses its internal randomness to initialize a random-tape for P_2, and executes P_2 on the given input until it blocks or halts. Then (using fresh randomness) it produces an authenticated ciphertext of the resulting configuration of P_2. It outputs this encrypted configuration along with the (unencrypted) contents of the output tape of P_2. P_1 can also be invoked with an encrypted configuration and an input: in this case, it checks the authentication, decrypts the configuration (which contains the random tape for P_2) and executes P_2 starting from this configuration, with the given input added to the input-tape.

The security of this reduction follows from the semantic security of the encryption and the existential unforgeability of the MAC.

B.2 General Construction for Interactive Agents

In this section, we shall reduce a general schema $\mathbf{\Sigma} = (\mathcal{P}_{\mathsf{auth}}, \mathcal{P}_{\mathsf{user}})$ to the schema $\mathbf{\Sigma}_R$ from Section B.1, which consists of arbitrary randomized (non-reactive, non-interactive) agents. Combined with the first of two reductions from the previous section, using Theorem 3, this gives a reduction of $\mathbf{\Sigma}$ to $\mathbf{\Sigma}_{\text{OBF}}$.

Our reduction $(\mathcal{O}, \mathcal{E})$ is fairly simple. At a high-level, $\mathcal{O}$ will upload an agent called P_{run} to $\mathcal{B}[\mathbf{\Sigma}_R]$, which will be used as a key for carrying out all sessions. The agents in the sessions are maintained as encrypted and authenticated configurations, which the P_{run} will decrypt, execute and update, and then reencrypt and sign. Note that for this P_{run} needs to be a randomized agent (hence the reduction to $\mathbf{\Sigma}_R$ rather than $\mathbf{\Sigma}_{\text{OBF}}$).

More precisely, during setup, $\mathcal{O}_{\mathsf{setup}}$ will pick a secret-key and public-key pair (SK, PK) for a CCA2-secure public-key encryption, and a pair of signing and verification keys (Sig, Ver) for a digital signature scheme, and sets $MPK = PK$ and $MSK = (SK, PK, Sig, Ver)$. It will also upload the following randomized agent P_{run} to $\mathcal{B}[\mathbf{\Sigma}_R]$: the parameters of P_{run} include MSK.

1. P_{run} takes as input $((C_1, \sigma_1, x_1), \cdots, (C_t, \sigma_t, x_t))$ for $t \geq 1$, where C_i are encrypted configurations of agents in $\mathcal{P}_{\mathsf{auth}} \cup \mathcal{P}_{\mathsf{user}}$, σ_i are signatures on C_i, and x_i are inputs for the agents.
2. It decrypts each C_i using SK. It also checks the signatures on all the ciphertexts using Ver, except for the ciphertexts that contain a start configuration[16] of an agent in $\mathcal{P}_{\mathsf{user}}$.
3. If all the configurations and signatures are valid, then first P_{run} chooses a seed for a pseudorandom

[16]A start configuration has all the tapes, except the parameter tape, empty. The configuration also contains information about the agent family that the agent belongs to.

generator (PRG) to define the random-tape of each agent in a start configuration.[17]

4. Then P_{run} copies the inputs x_i to input tapes of the respective agents and carries out a session execution.
5. When the session terminates, P_{run} encrypts each agent's configuration (along with the updated seed of the PRG that defines its random tape), to obtain ciphertexts C_i'; it signs them using Sig to get signatures σ_i'; finally, it halts after outputting $((C_1', \sigma_1', y_1), \cdots, (C_t', \sigma_t', y_t))$, where y_i are the contents of the output tapes of the agents.

After setup, when $\mathcal{O}_{\mathsf{auth}}$ is given an agent in $\mathcal{P}_{\mathsf{auth}}$ by Test, it simply encrypts (the start configuration of) the agent, signs it, and outputs the resulting pair (C, σ) as the obfuscation of the given agent. $\mathcal{O}_{\mathsf{user}}$ only encrypts the agent and outputs $(C, \perp)$ as the obfuscation; it is important that the encryption scheme used is CCA2 secure.

$\mathcal{E}$ behaves as follows. During setup, $\mathcal{E}$ receives PK from $\mathcal{O}$ and a handle from $\mathcal{B}[\mathbf{\Sigma}_R]$. $\mathcal{E}$ sends User the handles corresponding to agents which are uploaded by User, or received (as cryptographically encoded agents) from $\mathcal{O}$, or (as part of session output) from P_{run}. For each agent uploaded by User, $\mathcal{E}$ stores its parameters (i.e., start configuration) encrypted with PK, indexed by its handle. For cryptographically encoded agents received from $\mathcal{O}$ or P_{run}, it stores the obfuscation (C, σ), indexed by its handle. When given a session execution command, $\mathcal{E}$ retrieves the cryptographically encoded agents stored for each handle (with an empty signature if it is an agent in $\mathcal{P}_{\mathsf{user}}$ with start configuration) and sends them to $\mathcal{B}[\mathbf{\Sigma}_R]$, along with the handle for P_{run}. It gets back $((C_1', \sigma_1', y_1), \cdots, (C_t', \sigma_t', y_t))$ as the output from the session. It stores each (C_i', σ_i') received with a new handle, and sends these handles along with the outputs y_i to User.

The correctness of this reduction is straightforward, depending only on the security of the PRG (and only the correctness of the encryption and signature schemes). To prove the security property, we sketch a simulator $\mathcal{S}$. It internally runs $\mathcal{O}_{\mathsf{setup}}$ to produce (MSK, MPK) and sends the latter to Adv. It also sends Adv a handle, to simulate the handle for P_{run} it would receive during the setup phase. Subsequently, when $\mathcal{S}$ receives handles from $\mathcal{B}[\mathbf{\Sigma}]$ for agents uploaded by Test, it simulates the output of $\mathcal{O}_{\mathsf{auth}}$ or $\mathcal{O}_{\mathsf{user}}$ (depending on whether the handle is for $\mathcal{P}_{\mathsf{auth}}$ or $\mathcal{P}_{\mathsf{user}}$) by encrypting a dummy configuration for an agent. Note that the ciphertexts produced by $\mathcal{O}_{\mathsf{user}}$ are not signed. Also, when $\mathcal{S}$ receives new handles from a session executed by $\mathcal{B}[\mathbf{\Sigma}]$, it simulates the output of P_{run}, again by encrypting dummy configurations (these are signed ciphertexts). $\mathcal{S}$ hands over all such simulated ciphertexts to Adv, and also records them along with the corresponding handles it received from $\mathcal{B}[\mathbf{\Sigma}]$. When Adv sends a session execution command for P_{run}, with an input of the form $((C_1, \sigma_1, x_1), \cdots, (C_t, \sigma_t, x_t))$, $\mathcal{S}$ attempts to find a handle for each C_i as follows: first, $\mathcal{S}$ looks up the handles for the ciphertexts, if any, that it has recorded already. Note that if a ciphertext has a valid signature, it must have been generated and recorded by $\mathcal{S}$. But if there is any ciphertext which is not signed, and which does not appear in $\mathcal{S}$'s table, then $\mathcal{S}$ will decrypt the ciphertext; if that gives a valid start configuration for an agent in $\mathcal{P}_{\mathsf{user}}$, then $\mathcal{S}$ will upload that agent to $\mathcal{B}[\mathbf{\Sigma}]$, and obtains a handle for it. As we shall see, this is where the CCA2 security is crucial, as explained below. (If any of the above steps fail (invalid signature, invalid ciphertext or invalid decrypted configuration), $\mathcal{S}$ can simply simulate an empty output from P_{run}.) Once it has a handle h_i for every C_i, $\mathcal{S}$ asks $\mathcal{B}[\mathbf{\Sigma}]$ to execute a session with those handles and inputs $x_1, \cdots, x_t$. It returns the resulting outputs as well as dummy ciphertexts (as already described) as the output from P_{run}.

The proof that the simulation is good relies on the CCA2 security of the encryption scheme (as well as the unforgeability of the signatures, and the security of the PRG). Note that on obtaining handles for various agents from $\mathcal{B}[\mathbf{\Sigma}]$, $\mathcal{S}$ hands over dummy ciphertexts to Adv, and if Adv gives them back to

[17]We use the PRG in a stream-cipher mode: it produces a stream of pseudorandom bits, such that at any point there is an updated seed that can be used to continue extracting more bits from the PRG.

$\mathcal{S}$, it translates them back to the handles. Every other ciphertext is decrypted by $\mathcal{S}$ and used to create an agent that it uploads. However, if the encryption scheme were malleable, Adv could generate such a ciphertext by malleating one of the ciphertexts it received (from $\mathcal{S}$ or from, say, $\mathcal{O}_{\mathsf{user}}$). Thus in the real execution, the agent created by Adv would be related to an agent created by $\mathcal{O}_{\mathsf{user}}$, where as in the simulation it would be related to dummy agent created by $\mathcal{S}$, leading to a distinguishing attack. CCA2 security prevents this: one can translate a distinguishing attack (Test, Adv and $\mathcal{S}$ together) to an adversary in the CCA2 security experiment, in which, though the adversary does not have access to the decryption keys as $\mathcal{S}$ would, it can still carry out the decryptions carried out by $\mathcal{S}$ using the decryption oracle in the CCA2 experiment. The details of this reduction are fairly routine, and hence omitted.

C Obfuscation

C.1 Indistinguishability Obfuscation

Below, we provide a formal definition for indistinguishability obfuscation.

Definition 10 (Indistinguishability Obfuscation)**.** *A uniform* PPT *machine* $\textsc{obf}(\cdot)$ *is called an indistinguishability obfuscator for a circuit family* $\mathcal{F} = \{\mathcal{F}_\kappa\}$ *if it probabilistically maps circuits to circuits such that the following conditions are satisfied:*

- **Correctness:** $\forall \kappa \in \mathbb{N}$, $\forall C \in \mathcal{F}_\kappa$, *and* $\forall$ *inputs* x *we have that*
$$\Pr\left[C'(x) = C(x) : C' \leftarrow \textsc{obf}(1^\kappa, C)\right] = 1.$$
- **Relaxed Polynomial Slowdown:** *There exists a universal polynomial* p *such that for any circuit* C*, we have* $|C'| \leq p(|C|, \kappa)$ *where* $C' \leftarrow \textsc{obf}(1^\kappa, C)$.
- **Indistinguishability:** *For every pair of circuits* $C_0, C_1 \in \mathcal{F}_\kappa$*, such that* $\forall x,\ C_0(x) = C_1(x)$*, we have that for all* PPT *distinguishers* $\mathcal{D}$
$$\mathcal{D}(1^\kappa, \textsc{obf}(1^\kappa, C_0)) \approx \mathcal{D}(1^\kappa, \textsc{obf}(1^\kappa, C_1)).$$

Multiple obfuscated circuits: Using a hybrid argument, one can show that if $\textsc{obf}(\cdot)$ is an indistinguishability obfuscator, then security also holds against distinguishers who have access to multiple obfuscated circuits. More formally, let $(\mathcal{C}_0, \mathcal{C}_1)$ be a pair of sequence of circuits where $\mathcal{C}_b = \{C_{b,1}, C_{b,2}, \ldots, C_{b,\ell}\}$ for $b \in \{0,1\}$ (and ℓ is some polynomial in κ). Suppose for every $i \in [1,\ell]$ and for all x, $C_{0,i}(x) = C_{1,i}(x)$. Then, for all PPT distinguishers $\mathcal{D}$ we have that
$$\mathcal{D}(1^\kappa, \textsc{obf}(1^\kappa, C_{0,1}), \ldots, \textsc{obf}(1^\kappa, C_{0,\ell})) \approx \mathcal{D}(1^\kappa, \textsc{obf}(1^\kappa, C_{1,1}), \ldots, \textsc{obf}(1^\kappa, C_{1,\ell})).$$

C.1.1 Proof of Lemma 1

Proof. Suppose $(\mathcal{O}, \mathcal{E})$ is a set-up free Δ_{det}-IND-PRE-secure scheme for $\mathbf{\Sigma}_{\text{OBF}}$, then $\mathcal{O} \circ \mathcal{E}$ is an indistinguishability obfuscator, where $\mathcal{O} \circ \mathcal{E}$ is defined as discussed before. By construction, $\mathcal{O} \circ \mathcal{E}$ satisfies the correctness and polynomial slowdown requirements. So suppose $\mathcal{O} \circ \mathcal{E}$ does not satisfy the indistinguishability preservation property. Then there exists two circuits C_0 and C_1 which have identical input-output behavior, but there exists a PPT algorithm $\mathcal{D}$ which distinguishes between of $\mathcal{O} \circ \mathcal{E}(C_0)$ and $\mathcal{O} \circ \mathcal{E}(C_1)$. Now, define a simple $\mathsf{Test} \in \Delta_{\mathsf{det}}$ which on input b, uploads C_b. It is easy to see that Test is hiding w.r.t. $\mathbf{\Sigma}_{\text{OBF}}$ as the User gets only black-box access to C_0 or C_1. On the other hand, we argue that Test is not hiding w.r.t. $\mathcal{O}$. For this consider an adversary Adv which, on obtaining a string O from $\mathcal{O}$ constructs the program $Z := \mathcal{E}[O]$ and invokes $\mathcal{D}(Z)$. Then $\textsc{real}\langle\mathsf{Test}(0) \mid \mathcal{O} \mid \mathsf{Adv}\rangle \not\approx \textsc{real}\langle\mathsf{Test}(1) \mid \mathcal{O} \mid \mathsf{Adv}\rangle$ follows from the fact that Z is distributed as $\mathcal{O} \circ \mathcal{E}(C_b)$, where b is the input to Test, and the distinguishing advantage of $\mathcal{D}$.

We now show how $\textsc{obf}(.)$, an indistinguishability obfuscator, yields a (perfectly correct) set-up free Δ_{det}-IND-PRE-secure scheme for $\mathbf{\Sigma}_{\textsc{obf}}$. Together with the observation above, this will prove the lemma. We know that $\textsc{obf}$ maps circuits to circuits. Hence, $\mathcal{O}$ on input a circuit C runs $\textsc{obf}$ on the same input to obtain another circuit C'. This latter circuit is forwarded to $\mathcal{E}$. When $\mathcal{E}$ receives a circuit, it forwards a handle to the User; and when it receives a handle h and an input x from the User, it executes the circuit C' corresponding to h on x, and returns $C'(x)$. Correctness easily follows from the construction of $\mathcal{O}$ and $\mathcal{E}$.

Now, suppose that $(\mathcal{O}, \mathcal{E})$ is not a secure implementation. This implies that there exists a $\mathsf{Test} \in \Delta_{\mathsf{det}}$ which is hiding w.r.t $\mathbf{\Sigma}_{\textsc{obf}}$ but not w.r.t. $\mathcal{O}$. Hence, there exists an adversary Adv which can distinguish between $\mathsf{Test}(0)$ and $\mathsf{Test}(1)$ in the real world. Using Test and Adv, we construct a distinguisher $\mathcal{D}$ as follows. Recall that $\mathsf{Test}(b)$ can be represented as $\mathsf{D} \circ \mathfrak{cs}(b)$, where D is a deterministic party. $\mathcal{D}$ internally simulates a real world set-up with D, $\mathcal{O}$ and Adv. Note that every pair of circuits (C_0, C_1) that D sends to $\mathfrak{c}$ must be equivalent, otherwise Test would not be hiding w.r.t. $\mathbf{\Sigma}_{\textsc{obf}}$. When D uploads (C_0, C_1), $\mathcal{D}$ forwards them to Adv and the challenger. Let us say that the challenger picks a bit $b \in \{0, 1\}$. When $\mathcal{D}$ receives $\textsc{obf}(C_b) = \mathcal{O}(C_b)$ from the challenger, he forwards it to Adv. Finally, $\mathcal{D}$ outputs the view of the adversary. Since the view of Adv in the experiment where challenger picks b is identical to its view in the real world when Test has input b, $\mathcal{D}$ succeeds in distinguishing between the case where challenger picks 0 from the case where it picks 1. $\square$

C.2 Differing Inputs obfuscation.

Definition 11 (Differing Inputs Obfuscation)**.** *A uniform* PPT *machine* $\textsc{obf}(\cdot)$ *is called a differing inputs obfuscator for a circuit family* $\mathcal{F} = \{\mathcal{F}_\kappa\}$ *if it probabilistically maps circuits to circuits such that it satisfies the correctness and relaxed polynomial slowdown conditions as in Definition 10 and also:*

- **Differing Inputs:** *For every algorithm* $\mathsf{Sampler}$ *which takes* 1^κ *as input and outputs* (C_0, C_1, aux)*, where* $C_0, C_1 \in \mathcal{F}_\kappa$*, if for all* PPT $\mathcal{A}$

$$\Pr\left[C_0(x) \neq C_1(x) : (C_0, C_1, \mathsf{aux}) \leftarrow \mathsf{Sampler}(1^\kappa); x \leftarrow \mathcal{A}(1^\kappa, C_0, C_1, \mathsf{aux})\right] \leq \mathrm{negl}(\kappa),$$

then for all PPT *distinguishers* $\mathcal{D}$

$$\mathcal{D}(1^\kappa, \textsc{obf}(1^\kappa, C_0), \mathsf{aux}) \approx \mathcal{D}(1^\kappa, \textsc{obf}(1^\kappa, C_1), \mathsf{aux}).$$

We call the $\mathsf{Sampler}$ whose output satisfies the condition given above (against all PPT $\mathcal{A}$) a *good* sampler. Note that differing inputs obfuscation requires indistinguishability to hold for good samplers only.

Multiple obfuscated circuits : Like in the case of indistinguishability obfuscation, we can show that if a differing inputs obfuscator $\textsc{obf}(\cdot)$ exists, then it is also secure against the following more general class of sampling functions. Let $\mathsf{Sampler}_\ell$ be an algorithm that on input 1^κ outputs a pair of sequence of circuits $(\mathcal{C}_0, \mathcal{C}_1)$ and aux, where $\mathcal{C}_b = \{C_{b,1}, C_{b,2}, \ldots, C_{b,\ell}\}$ for $b \in \{0, 1\}$ (and ℓ is some polynomial in κ). We claim that if $\mathsf{Sampler}_\ell$ is *good*, i.e., for all PPT $\mathcal{A}$,

$$\Pr\left[C_{0,i}(x) \neq C_{1,i}(x) : (\mathcal{C}_0, \mathcal{C}_1, \mathsf{aux}) \leftarrow \mathsf{Sampler}(1^\kappa); (x, i) \leftarrow \mathcal{A}(1^\kappa, \mathcal{C}_0, \mathcal{C}_1, \mathsf{aux})\right] \leq \mathrm{negl}(\kappa),$$

then for all PPT distinguishers $\mathcal{D}$,

$$\mathcal{D}(1^\kappa, \textsc{obf}(1^\kappa, C_{0,1}), \ldots, \textsc{obf}(1^\kappa, C_{0,\ell}), \mathsf{aux}) \approx \mathcal{D}(1^\kappa, \textsc{obf}(1^\kappa, C_{1,1}), \ldots, \textsc{obf}(1^\kappa, C_{1,\ell}), \mathsf{aux}).$$

We can prove this claim via a hybrid argument. For $i \in [0, \ell]$, let $\mathcal{H}_i$ be the hybrid consisting of $\textsc{obf}(C_{1,1}), \ldots, \textsc{obf}(C_{1,i})$, $\textsc{obf}(C_{0,i+1}), \ldots, \textsc{obf}(C_{0,\ell})$ and aux (κ has been omitted for convenience). In order to show that $\mathcal{H}_0$ is indistinguishable from $\mathcal{H}_\ell$, it is sufficient to show that for every $i \in [0, \ell{-}1]$, $\mathcal{H}_i$ is indistinguishable from $\mathcal{H}_{i+1}$. Both $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ have $\textsc{obf}(C_{1,1}), \ldots, \textsc{obf}(C_{1,i})$, $\textsc{obf}(C_{0,i+2}), \ldots, \textsc{obf}(C_{0,\ell})$

and aux in common. The only difference is that while the former has $\text{OBF}(C_{0,i+1})$, the latter has $\text{OBF}(C_{1,i+1})$.

Consider an algorithm $\mathsf{Sampler}$ which on input 1^κ, runs $\mathsf{Sampler}_\ell(1^\kappa)$ and outputs $(C_{0,i+1}, C_{1,i+1}, \mathsf{aux}')$, where $\mathsf{aux}' = (\mathcal{C}_0, \mathcal{C}_1, \mathsf{aux})$. We can easily show that $\mathsf{Sampler}$ is a good sampling algorithm if $\mathsf{Sampler}_\ell$ is good. Hence, $(\text{OBF}(C_{0,i+1}), \mathsf{aux}')$ is indistinguishable from $(\text{OBF}(C_{1,i+1}), \mathsf{aux}')$. This implies that $\mathcal{H}_i$ is indistinguishable from $\mathcal{H}_{i+1}$.

C.2.1 Proof of Lemma 2

Proof. We first show the only if direction. Let $(\mathcal{O}, \mathcal{E})$ be a Δ^*-IND-PRE-secure scheme for $\mathbf{\Sigma}_{\text{OBF}}$. We claim that $\mathcal{O} \circ \mathcal{E}$ is a differing inputs obfuscator, where $\mathcal{O} \circ \mathcal{E}$ is defined as discussed before. Towards this, let $\mathcal{S}$ be a good sampling algorithm which takes 1^κ as input and outputs (C_0, C_1, aux). We define a $\mathsf{Test}_\mathcal{S} \in \Delta^*$ as follows: D runs $\mathcal{S}$ to obtain (C_0, C_1, aux); it sends aux to the User and (C_0, C_1) to $\mathfrak{c}$. The only way an adversary can distinguish between the case where $\mathfrak{s}$ uploads C_0 from the case where it uploads C_1 is if it queries $\mathcal{B}[\mathbf{\Sigma}_{\text{OBF}}]$ with an input x s.t. $C_0(x) \neq C_1(x)$. But this is not possible because $\mathcal{S}$ is a good sampler. Therefore, $\mathsf{Test}_\mathcal{S}$ is hiding w.r.t. $\mathbf{\Sigma}_{\text{OBF}}$. This implies that $\mathsf{Test}_\mathcal{S}$ is hiding w.r.t. $\mathcal{O}$ as well. It is now easy to show that $(\mathcal{O} \circ \mathcal{E}(C_0), \mathsf{aux})$ is indistinguishable from $(\mathcal{O} \circ \mathcal{E}(C_1), \mathsf{aux})$.

We now show the if direction of the lemma. Suppose $\text{OBF}(\cdot)$ is a differing inputs obfuscator. Using OBF we can define a scheme $(\mathcal{O}, \mathcal{E})$ in the natural way (see the proof of the previous lemma for details). We claim that $(\mathcal{O}, \mathcal{E})$ is an Δ^*-IND-PRE-secure scheme for $\mathbf{\Sigma}_{\text{OBF}}$. Correctness easily follows from construction. In order to prove indistinguishability preservation, consider a $\mathsf{Test} \in \Delta^*$. Let $(\mathcal{C}_0, \mathcal{C}_1)$ denote the sequence of pairs of circuits uploaded by D, where $\mathcal{C}_b = \{C_{b,1}, C_{b,2}, \ldots, C_{b,\ell}\}$ for $b \in \{0, 1\}$, and aux be the messages sent to the User. Observe that for both $b = 0$ and 1, any adversary Adv receives $\mathcal{C}_0, \mathcal{C}_1, \mathsf{aux}$, and a sequence of handles $h_1, \ldots, h_\ell$. If Test is hiding w.r.t $\mathbf{\Sigma}_{\text{OBF}}$, then the probability that Adv queries with h_i and input x such that $C_{0,i}(x) = C_{1,i}(x)$ is negligible. Hence an algorithm $\mathsf{Sampler}$ which runs Test and outputs $(\mathcal{C}_0, \mathcal{C}_1, \mathsf{aux})$ is a good sampling algorithm. Therefore, $(\text{OBF}(C_{0,1}), \ldots, \text{OBF}(C_{0,\ell}), \mathsf{aux})$ cannot be distinguished from $(\text{OBF}(C_{1,1}), \ldots, \text{OBF}(C_{1,\ell}), \mathsf{aux})$. We can now show that Test is hiding w.r.t. $\mathcal{O}$ in a manner similar to the previous lemma. ☐

D Functional Encryption

D.1 Traditional Definition of Functional Encryption

The following definition of functional encryption is from [21, 71]. It corresponds to non-function-hiding, public-key functional encryption.

Syntax. A functional encryption scheme $\mathcal{FE}$ for a circuit family $\mathcal{F} = \{\mathcal{F}_\kappa\}$ over a message space $\mathcal{X} = \{\mathcal{X}_\kappa\}$ consists of four PPT algorithms:

- $\mathsf{Setup}(1^\kappa)$ takes as input the unary representation of the security parameter, and outputs the master public and secret keys $(\mathsf{MPK}, \mathsf{MSK})$;
- $\mathsf{KeyGen}(\mathsf{MSK}, C)$ takes as input the master secret key MSK and a circuit $C \in \mathcal{F}_\kappa$, and outputs a corresponding secret key SK_C;
- $\mathsf{Encrypt}(\mathsf{MPK}, x)$ takes as input the master public key MPK and a message $x \in \mathcal{X}_\kappa$, and outputs a ciphertext CT_x;
- $\mathsf{Decrypt}(\mathsf{SK}_C, \mathsf{CT}_x)$ takes as input a key SK_C and a ciphertext CT_x, and outputs a value.

These algorithms must satisfy the following *correctness* property for all $\kappa \in \mathbb{N}$, all $C \in \mathcal{F}_\kappa$ and all $x \in \mathcal{X}_\kappa$,

$$\Pr\left[\begin{array}{c}(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\kappa); \\ \quad \mathsf{Decrypt}(\mathsf{KeyGen}(\mathsf{MSK}, C), \mathsf{Encrypt}(\mathsf{MPK}, x)) \neq C(x)\end{array}\right] = \text{negl}(\kappa),$$

where the probability is taken over their coin tosses.

Indistinguishability Security. The standard indistinguishability based security definition for functional encryption is defined as a game between a challenger and an adversary $\mathcal{A}$ as follows.

- **Setup**: The challenger runs $\mathsf{Setup}(1^\kappa)$ to obtain $(\mathsf{MPK}, \mathsf{MSK})$, and gives MPK to $\mathcal{A}$.
- **Key queries**: $\mathcal{A}$ sends a circuit $C \in \mathcal{C}_\kappa$ to the challenger, and receives $\mathsf{SK}_C \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, C)$ in return. This step can be repeated any polynomial number of times.
- **Challenge**: $\mathcal{A}$ submits two messages x_0 and x_1 such that $C(x_0) = C(x_1)$ for all C queried by $\mathcal{A}$ in the previous step. Challenger sends $\mathsf{Encrypt}(\mathsf{MPK}, x_b)$ to $\mathcal{A}$.
- **Adaptive key queries**: $\mathcal{A}$ continues to send circuits to the challenger subject to the restriction that any C queried must satisfy $C(x_0) = C(x_1)$.
- **Guess**: $\mathcal{A}$ outputs a bit b'.

The advantage of $\mathcal{A}$ in this security game is given by $|\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]|$. We say that a functional encryption scheme $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is *indistinguishability secure* if for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the security game described above is negligible in κ.

Multiple challenge phases : One can show via a hybrid argument that if no adversary has a significant advantage in the above security game, then the same holds for a generalized game where there are multiple challenge phases interspersed with key query phases. In the generalized game, it is required that for every (x_0, x_1) submitted in a challenge phase, and every circuit C queried in any key query phase, $C(x_0) = C(x_1)$.

D.2 Proof of Lemma 3

Proof. We first prove the easier side. Let $(\mathcal{O}, \mathcal{E})$ be a Δ_{det}-IND-PRE-secure scheme for $\mathbf{\Sigma}_{\text{FE}}$, where $\mathcal{O} = (\mathcal{O}_{\mathsf{setup}}, \mathcal{O}_{\mathsf{auth}}, \mathcal{O}_{\mathsf{user}})$. We construct an FE scheme $\mathbb{S}_{\text{FE}}$ using $(\mathcal{O}, \mathcal{E})$ as follows.

- $\mathsf{Setup}(1^\kappa)$**:** Run $\mathcal{O}_{\mathsf{setup}}$ to obtain a master secret key MSK and a public key MPK.
- $\mathsf{KeyGen}(\mathsf{MSK}, C)$**:** Output $\mathsf{SK}_C \leftarrow \mathcal{O}_{\mathsf{auth}}(C; \mathsf{MSK})$ (where C is passed to $\mathcal{O}_{\mathsf{auth}}$ as the parameter for the agent $P_C^{\mathsf{Fun}} \in \mathcal{P}_{\mathsf{auth}}^{\mathsf{PubFE}}$).
- $\mathsf{Encrypt}(\mathsf{MPK}, x)$**:** Output $\mathsf{CT}_x \leftarrow \mathcal{O}_{\mathsf{user}}(x; \mathsf{MPK})$ (where x is passed to $\mathcal{O}_{\mathsf{user}}$ as the parameter for the agent $P_x^{\mathsf{Msg}} \in \mathcal{P}_{\mathsf{user}}^{\mathsf{PubFE}}$).
- $\mathsf{Decrypt}(\mathsf{SK}_C, \mathsf{CT}_x)$**:** Run a copy of $\mathcal{E}$ as follows: first feed it SK_C and CT_x as messages from $\mathcal{O}$, and obtain agent handles h_C and h_x; then request it for a session execution with handles (h_C, h_x) (and no input). Return the output for the agent h_C as reported by $\mathcal{E}$.

In order to show that $\mathbb{S}_{\text{FE}}$ is an indistinguishability secure FE scheme, we consider the following $\mathsf{Test} \in \Delta_{\mathsf{det}}$. Upon receipt of a circuit C from User, Test uploads C and adds C to a list L. Upon receipt of a pair of inputs (x_0, x_1), if for every $C \in L$, $C(x_0) = C(x_1)$, Test uploads x_b. After this, if User sends a circuit C', Test uploads C' iff $C'(x_0) = C'(x_1)$. (If User sends any other type of message, it is ignored.)

Now suppose there is an adversary $\mathcal{A}$ who breaks the security of $\mathbb{S}_{\text{FE}}$. Then we show that the above Test is hiding w.r.t. $\mathbf{\Sigma}_{\text{FE}}$ but not w.r.t. $\mathcal{O}$. To see this, firstly note that Test is hiding w.r.t. $\mathbf{\Sigma}_{\text{FE}}$ by design: there is no way an adversary can learn whether Test uploaded x_0 or x_1 in the ideal world. Now, consider an adversary Adv who runs $\mathcal{A}$ internally: first it forwards MPK received from $\mathcal{O}_{\mathsf{setup}}$ to $\mathcal{A}$; then it forwards $\mathcal{A}$'s requests to the challenger (in the IND security game) to Test; the outputs received from $\mathcal{O}$ are forwarded to $\mathcal{A}$. Finally Adv outputs $\mathcal{A}$'s output bit. It is straightforward to see that the advantage Adv has in distinguishing interaction with $\mathsf{Test}(0)$ and $\mathsf{Test}(1)$ is exactly the advantage $\mathcal{A}$ has in the IND security experiment.

We now prove the other side of the lemma. Let $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be an indistinguishability secure FE scheme. We can construct a Δ_{det}-$\mathsf{IND\text{-}PRE}$-secure scheme $(\mathcal{O}, \mathcal{E})$ for $\mathbf{\Sigma}_{\text{FE}}$ using the scheme $\mathcal{FE}$ in a way analogous to how an IND secure FE scheme is constructed from an $\mathsf{IND\text{-}PRE}$ secure scheme above. We now show that if $(\mathcal{O}, \mathcal{E})$ is not a secure scheme then neither is $\mathcal{FE}$. That is, if there exists a $\mathsf{Test} \in \Delta_{\mathsf{det}}$ such that Test is hiding in the ideal world, but there exists a PPT adversary Adv which can distinguish between $\mathsf{Test}(0)$ and $\mathsf{Test}(1)$ in the real world, then there exists an adversary $\mathcal{A}$ which can break the security of $\mathcal{FE}$ in the generalized IND game.

Recall that $\mathsf{Test}(b)$ can be represented as $\mathsf{D} \circ \mathfrak{cs}(b)$, where D is a deterministic party. $\mathcal{A}$ internally simulates a real world set-up with D, $\mathcal{O}$ and Adv, and externally participates in the indistinguishability game with a challenger. We will show that for $b \in \{0, 1\}$, if challenger picks the bit b, then Adv's view is identically distributed to its view in the real world when Test gets input b. This will complete the proof.

At any point during a run of the real world, D either uploads a pair of function agents (C_0, C_1) or a pair of message agents (x_0, x_1) to $\mathfrak{c}$. We can see that C_0 and C_1 must be the same circuits, otherwise Test would not be hiding in the ideal world. Similarly, for every function agent $C = C_0 = C_1$ ever uploaded, it must be the case that $C(x_0) = C(x_1)$, for every (x_0, x_1). It is now easy to simulate the view of Adv. When a function agent C is uploaded by D, $\mathcal{A}$ sends C to the challenger, and forwards the key obtained to Adv (along with (C_0, C_1)). When D uploads (x_0, x_1), $\mathcal{A}$ forwards it to the challenger. The ciphertext returned by the challenger is forwarded to Adv (along with (x_0, x_1)). $\square$

E Fully Homomorphic Encryption

Given a Δ_{det}-$\mathsf{IND\text{-}PRE}$ secure scheme $(\mathcal{O}, \mathcal{E})$ for $\mathbf{\Sigma}_{\text{FHE}}$, we show how to construct a semantically secure FHE scheme $\mathbb{S}_{\text{FHE}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Eval})$. For a formal treatment of FHE, see [53].

- $\mathsf{Setup}(1^\kappa)$: Run $\mathcal{O}_{\mathsf{setup}}$ to obtain public key PK and secret key SK.
- $\mathsf{Encrypt}(x, \mathsf{PK})$: Run $\mathcal{O}_{\mathsf{user}}((0, x), \mathsf{PK})$ to obtain a ciphertext CT_x. Here 0 denotes that x is a parameter for agent P^{Msg}.
- $\mathsf{Decrypt}(\mathsf{CT}, \mathsf{SK})$: Let $D \leftarrow \mathcal{O}_{\mathsf{auth}}(1, \mathsf{SK})$, where 1 denotes that the agent is P^{Dec}. Then run a copy of $\mathcal{E}$ as follows: first feed it D and CT as messages from $\mathcal{O}$, and obtain handles h_D and h_m; then request it for a session execution with $(h_D, \perp)$ and $(h_m, \perp)$. Return the output for the agent h_D as reported by $\mathcal{E}$.
- $\mathsf{Eval}(C, \mathsf{CT}_1, \mathsf{CT}_2, \ldots, \mathsf{CT}_n)$: Run a copy of $\mathcal{E}$ as follows: first feed it $\mathsf{CT}_1, \mathsf{CT}_2, \ldots, \mathsf{CT}_n$ as messages from $\mathcal{O}$, and obtain handles $h_1, h_2, \ldots, h_n$. Then request $\mathcal{E}$ to run a session with (h_1, f), $(h_2, \perp)$, $\ldots, (h_n, \perp)$. Output the ciphertext CT returned by $\mathcal{E}$.

Correctness follows easily from construction. Compactness follows from the fact that the size of string recorded for each handle by $\mathcal{E}$ is *a priori* bounded. We now show that $\mathbb{S}_{\text{FHE}}$ is semantically secure. On the contrary, suppose there exists an adversary $\mathcal{A}$ who breaks the semantic security of $\mathbb{S}_{\text{FHE}}$. Consider the following $\mathsf{Test}(b)$: Upon receipt of inputs x_0, x_1 from User, Test chooses x_b and uploads it (as parameter for agent P^{Msg}). This Test is clearly hiding in the ideal world, because in the absence of a decryption agent, an adversary only obtains handles from $\mathcal{B}[\mathbf{\Sigma}_{\text{FHE}}]$. Therefore, by $\mathsf{IND\text{-}PRE}$ security of $(\mathcal{O}, \mathcal{E})$, Test is also hiding w.r.t. $\mathcal{O}$.

Now, consider an adversary Adv who runs $\mathcal{A}$ internally: first it forwards PK received from $\mathcal{O}_{\mathsf{setup}}$ to $\mathcal{A}$; then it forwards $\mathcal{A}$'s requests (x_0, x_1) to the challenger (in the semantic security game) to Test; the outputs received from $\mathcal{O}$ are forwarded to $\mathcal{A}$. Finally Adv outputs $\mathcal{A}$'s output bit. It is straightforward to see that the advantage Adv has in distinguishing interaction with $\mathsf{Test}(0)$ and $\mathsf{Test}(1)$ is exactly the advantage $\mathcal{A}$ has in the semantic security experiment.

F Property Preserving Encryption

F.1 Definitions

In this section, we formally define PPE and the standard notion of security for it [72].

Definition 12 (PPE scheme)**.** *A property preserving encryption scheme for a binary property $P : \mathcal{M} \times \mathcal{M} \to \{0,1\}$ is a tuple of four* PPT *algorithms defined as follows:*

- $\mathsf{Setup}(1^\kappa)$ *takes as input the security parameter κ and outputs a secret key* SK *(and some public parameters).*
- $\mathsf{Encrypt}(m, \mathsf{SK})$ *takes as input a message $m \in \mathcal{M}$ and outputs a ciphertext* CT*.*
- $\mathsf{Decrypt}(\mathsf{CT}, \mathsf{SK})$ *takes as input a ciphertext* CT *and outputs a message $m \in \mathcal{M}$.*
- $\mathsf{Test}(\mathsf{CT}_1, \mathsf{CT}_2)$ *takes as input two ciphertexts* CT_1 *and* CT_2 *and outputs a bit b.*

We require that for all messages $m, m_1, m_2 \in \mathcal{M}$, the following two conditions hold:

- *Decryption:* $\Pr[\mathsf{SK} \leftarrow \mathsf{Setup}(1^\kappa); \mathsf{Decrypt}(\mathsf{Encrypt}(m, \mathsf{SK}), \mathsf{SK}) \neq m] = \mathsf{negl}(\kappa)$*, and*
- *Property testing:* $\Pr[\mathsf{SK} \leftarrow \mathsf{Setup}(1^\kappa); \mathsf{Test}(\mathsf{Encrypt}(m_1, \mathsf{SK}), \mathsf{Encrypt}(m_2, \mathsf{SK})) \neq P(m_1, m_2)] = \mathsf{negl}(\kappa)$,

where the probability is taken over the random choices of the four algorithms.

Security. In [72], the authors show that there exists a hierarchy of meaningful indistinguishability based security notions for PPE, which does not collapse unlike other familiar settings. At the top of the hierarchy lies *Left-or-Right* (LoR) security, a notion that is similar to full security in symmetric key functional encryption.

LoR security. Let $\Pi_P = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Test})$ be a PPE scheme for a binary property P. Consider an adversary $\mathcal{A}$ in the security game $\exp^{(b)}_{\mathsf{LoR},\mathcal{A}}(1^\kappa)$ described below, for $b \in \{0,1\}$. The Setup algorithm is run to obtain a secret key SK and some public parameters. $\mathcal{A}$ is given the parameters, and access to an oracle $\mathcal{O}_b(\mathsf{SK}, \cdot, \cdot)$, such that $\mathcal{O}_b(\mathsf{SK}, m_0, m_1) = \mathsf{Encrypt}(m_b, \mathsf{SK})$. Let $Q = \{(m_1^{(0)}, m_1^{(1)}), (m_2^{(0)}, m_2^{(1)}), \ldots, (m_\ell^{(0)}, m_\ell^{(1)})\}$ denote the queries made by $\mathcal{A}$ to the oracle. At the end of the experiment, $\mathcal{A}$ produces an output bit; let this be the output of the experiment. We call $\mathcal{A}$ admissible if for every two (not necessarily distinct) pairs of messages $(m_i^{(0)}, m_i^{(1)}), (m_j^{(0)}, m_j^{(1)}) \in Q$, $P(m_i^{(0)}, m_j^{(0)}) = P(m_i^{(1)}, m_j^{(1)})$. We also refer to such messages as admissible.

Definition 13 (LoR security)**.** *The scheme Π_P is an* LoR *secure* PPE *scheme for a property P if for all* PPT *admissible adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ defined as below is negligible in the security parameter κ:*

$$\mathsf{Adv}_{\mathsf{LoR},\mathcal{A}}(\kappa) := \left| \Pr[\exp^{(0)}_{\mathsf{LoR},\mathcal{A}}(1^\kappa) = 1] - \Pr[\exp^{(1)}_{\mathsf{LoR},\mathcal{A}}(1^\kappa) = 1] \right|,$$

where the probability is over the random coins of the algorithms of Π_P and that of $\mathcal{A}$.

F.2 Equivalence

In this section, we show that Δ_{det}-IND-PRE and LoR security notions are equivalent for PPE.

Theorem 7. *A Δ_{det}-IND-PRE secure scheme for $\mathbf{\Sigma}_{\mathsf{PPE}}$ exists if and only if an* LoR *secure scheme for PPE exists.*

We first prove the only if side of the theorem. Let $(\mathcal{O}, \mathcal{E})$ be a Δ_{det}-IND-PRE-secure scheme for $\mathbf{\Sigma}_{\mathsf{PPE}}$, where $\mathcal{O} = (\mathcal{O}_{\mathsf{setup}}, \mathcal{O}_{\mathsf{auth}}, \mathcal{O}_{\mathsf{user}})$. We construct a PPE scheme $\mathbb{S}_{\mathsf{PPE}}$ using $(\mathcal{O}, \mathcal{E})$ as follows.

- $\mathsf{Setup}(1^\kappa)$: Run $\mathcal{O}_{\mathsf{setup}}$ to obtain the public parameters MPK and secret key MSK.
- $\mathsf{Encrypt}(m, \mathsf{MSK})$: Output $CT_m \leftarrow \mathcal{O}_{\mathsf{auth}}((0, m), \mathsf{MSK})$ where the first bit 0 indicates that the parameter m is for the agent P^{Msg}.
- $\mathsf{Decrypt}(\mathsf{CT}, \mathsf{MSK})$: Let $D \leftarrow \mathcal{O}_{\mathsf{auth}}(1, \mathsf{MSK})$, where 1 denotes that the agent is P^{Dec}. Then run a copy of $\mathcal{E}$ as follows: first feed it D and CT as messages from $\mathcal{O}$, and obtain handles h_D and h_m; then request it for a session execution with $(h_D, \perp)$ and (h_m, send). Return the output for the agent h_D as reported by $\mathcal{E}$.
- $\mathsf{Test}(\mathsf{CT}_1, \mathsf{CT}_2)$: Run a copy of $\mathcal{E}$ as follows: first feed it CT_1 and CT_2 as messages from $\mathcal{O}$, and obtain handles h_1 and h_2. Then request $\mathcal{E}$ to run a session with $(h_1, \mathsf{compute})$ and (h_2, send). Output the answer returned by $\mathcal{E}$.

It is easy to see that the decryption and property testing properties of PPE are satisfied. In order to show that $\mathbb{S}_{\mathrm{FE}}$ is an LoR secure PPE scheme, we consider the following $\mathsf{Test} \in \Delta_{\mathsf{det}}$. Let L be a list of pairs of messages, which is initially empty. Upon receipt of a pair (m_0, m_1) from User, Test checks if for every $(m_0', m_1') \in L$, $\mathsf{Test}(m_0, m_0') = \mathsf{Test}(m_1, m_1')$ and vice versa, and $\mathsf{Test}(m_0, m_0) = \mathsf{Test}(m_1, m_1)$. If the checks pass, Test uploads m_b and adds (m_0, m_1) to the list; otherwise this pair is ignored. (If Test receives a single message m from the User, it is treated as a pair (m, m).) Now suppose there is an adversary $\mathcal{A}$ who breaks the security of $\mathbb{S}_{\mathsf{PPE}}$. Then we can show that the above Test is hiding w.r.t. $\mathbf{\Sigma}_{\mathsf{PPE}}$ but not w.r.t. $\mathcal{O}$. The proof is very similar to the one given for Lemma 3, so we omit it here.

Next, we prove the if side of the theorem. Let $\mathbb{S}_{\mathsf{PPE}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Test})$ be an LoR secure PPE scheme. We construct a scheme $(\mathcal{O}, \mathcal{E})$ in the cryptographic agents framework as follows:

- $\mathcal{O}_{\mathsf{setup}}(1^\kappa)$**:** Run $\mathsf{Setup}(1^\kappa)$ to obtain $(\mathsf{MPK}, \mathsf{MSK})$.
- $\mathcal{O}_{\mathsf{auth}}((b, m); \mathsf{MSK})$**:** If $b = 0$, output $\mathsf{CT} \leftarrow \mathsf{Encrypt}(m, \mathsf{MSK})$, else output MSK itself. (Recall that 0 denotes an agent in P^{Msg}, while 1 denotes an agent in P^{Dec}.)
- $\mathcal{E}$**:** When $\mathcal{O}$ sends a ciphertext CT, forward a handle h to the User and store (h, CT). When $\mathcal{O}$ sends a key MSK, forward the handle h_{key} and store $(h_{\mathsf{key}}, \mathsf{MSK})$. When User requests a session execution with $(h_1, \mathsf{compute})$ and (h_2, send), retrieve the corresponding ciphertexts CT_1 and CT_2, and return $\mathsf{Test}(\mathsf{CT}_1, \mathsf{CT}_2)$ to the User. On the other hand, when User sends $(h_{\mathsf{key}}, \perp)$ and (h, send), retrieve the ciphertext CT corresponding to h, and return $\mathsf{Decrypt}(\mathsf{CT}, \mathsf{MSK})$ to the User.

We now show that if $(\mathcal{O}, \mathcal{E})$ is not a secure scheme then neither is $\mathbb{S}_{\mathsf{PPE}}$. That is, if there exists a $\mathsf{Test} \in \Delta_{\mathsf{det}}$ such that Test is hiding w.r.t. $\mathbf{\Sigma}_{\mathsf{PPE}}$ but not w.r.t. $\mathcal{O}$, then there exists an adversary $\mathcal{A}$ which can break the security of $\mathbb{S}_{\mathsf{PPE}}$ in the LoR security game. Recall that $\mathsf{Test}(b)$ can be represented as $\mathsf{D} \circ \mathfrak{cs}(b)$, where D is a deterministic party. It is clear that if D ever uploads a key agent, then every pair of messages (m_0, m_1) that it uploads must be such that $m_0 = m_1$ (otherwise Test would not be hiding in the ideal world). Such a Test would trivially be hiding in the real world. On the other hand, even if D never uploads a key agent, it must always upload admissible pairs of messages (see definition of LoR security) to remain hiding w.r.t. $\mathbf{\Sigma}_{\mathsf{PPE}}$. Rest of the proof is similar to Lemma 3, and hence omitted.