Receiver and Sender Deniable Functional Encryption

Angelo De Caro[1], Vincenzo Iovino[2*], Adam O'Neill[3]

[1]IBM Research, Zurich
[2]University of Luxembourg
[3]Georgetown University
[*]vinciovino@gmail.com

Abstract: *Deniable encryption*, first introduced by Canetti *et al.* (CRYPTO 1997), allows equivocation of encrypted communication. In this work we generalize its study to *functional encryption* (FE). Our results are summarized as follows:

We first put forward and motivate the concept of receiver deniable FE, for which we consider two models. In the first model, as previously considered by O'Neill *et al.* (CRYPTO 2011) in the case of identity-based encryption, a receiver gets assistance from the master authority to generate a fake secret key. In the second model, there are "normal" and "deniable" secret keys, and a receiver in possession of a deniable secret key can produce a fake but authentic-looking normal key on its own.

In the first model, we show a compiler from any FE scheme for the general circuit functionality to a FE scheme having receiver deniability. In addition we show an efficient receiver deniable FE scheme for Boolean Formulae from bilinear maps. In the second (multi-distributional) model, we present a specific FE scheme for the general circuit functionality having receiver deniability. To our knowledge, a scheme in the multi-distributional model was not previously known even for the special case of identity-based encryption.

Finally, we construct the first sender (non-multi-distributional) deniable FE scheme.

1. Introduction

Deniable encryption, formalized by Canetti *et al.* in 1997 [19], allows the sender and/or receiver of encrypted communication, after having already exchanged an encrypted message, to produce fake but authentic-looking random coins that "open" the ciphertext to a different message. Canetti *et al.* distinguish between two different models of deniability. The first is *deniability*, in which the parties always run the prescribed key generation and encryption algorithms, and can equivocate their messages later on. The second model is called *multi-distributional deniability*, in which there exist alternative *deniable algorithms* whose outputs can be equivocated, so that it appears as if the prescribed algorithms had been used all along. O'Neill *et al.* [40] construct "bideniable" public-key encryption schemes, namely where the sender and receiver can simultaneously equivocate without any coordination, albeit in the multidistributional model. O'Neill *et al.* also present a bideniable construction for identity-based encryption [42, 13] in which the receiver needs assistance from the master authority to produce fake secret keys.

Deniability for Functional Encryption. In this work, we generalize the study of deniability to *functional encryption* (FE) [15]. Deniability for identity-based encryption, that is a very special

case of FE, was also previously considered by [40]. Whereas traditional public-key encryption decryption is an all-or-nothing affair (*i.e.*, a receiver is either able to recover the entire message using its key, or nothing), in FE is possible to finely control the amount of information revealed by a ciphertext to a given receiver. Precisely, in a FE scheme for functionality F, each secret key (generated by a master authority) is associated with some value k. Anyone can encrypt via the public parameters. When a ciphertext Ct_x that encrypts x is decrypted using a secret key Sk_k for value k, the result is $F(k, x)$. Intuitively, security requires that a receiver in possession of Sk_k learns nothing beyond $F(k, x)$. We motivate the importance of deniability in the context of FE. For example, consider an encrypted email server that uses FE, where the server is in possession of keys that allow it to do searches, spam filtering, targeted advertising, *etc.* If the government coerces the email server to provide its keys or requests additional keys from the client, the server can do so in a way that again reveals any apparent values it likes. This is pertinent in light of recent political events and the closing of several email providers that did not want to comply with NSA demands (Lavabit and Silent Circle). As another scenario, consider a secure routing protocol implemented by means of FE, where each node receives an encrypted packet and using a secret key corresponding to its routing table is able to forward the packet to the right port without knowing the next destinations of the packet. The NSA could attempt to coerce the nodes to reveal their respective routing tables to trace the final destinations of the packet. If the FE system is receiver deniable, there is no rationale for the NSA to coerce a node as a node could just reveal a fake secret key.

Model and Definitions. Firstly, we put forth the concept of *receiver-deniable FE* and later, we will introduce *sender deniability*. For intuition, suppose the coercer has observed a ciphertext Ct_x that encrypts some string x. Informally, receiver-deniable FE allows the sender to produce "fake" secret key Sk'_k that makes equivocate Ct_x as encryption of any other x' so that the fake secret key decrypts to $F(k, x')$. But this intuition for the definition hides several details. First, what if the coercer is able to coerce many receivers, thus seeing many secret keys? In the case of identity-based encryption, it was previously observed by O'Neill *et al.* [40] that this case is equivalent via a hybrid argument to equivocation of a single ciphertext and secret key. However, this hybrid argument fails in our more general setting. Therefore, in our modeling we consider what we call (n_c, n_k)-*receiver-deniability*, denoting that the coercer requests n_c challenge ciphertexts and n_k secret keys (i.e, receiver-coerce queries) adaptively. Note that what we have described above is the "full", rather than "multi-distributional", model of deniability. While this avoids the need for the receiver to run special "deniable" algorithms in order to equivocate, it does mean that we need to assume the receiver can get assistance from the master authority to generate a fake secret key Sk'_k, and that the latter cannot be coerced (*e.g.,* because it is geographically outside the jurisdiction of the coercer). Note that previous works treated IBE in this model due to the belief that coordination with master authority was *necessary* to obtain receiver deniability.

Surprisingly, we show that this belief is not correct. In the multi-distributional model, (which is incomparable the full model), receiver-deniability can be obtained. We present a formal security definition for receiver-deniability in the multi-distributional model. In this model, the receiver needs to run special "deniable" algorithms in order to equivocate. The benefit is that the receiver does not need to get assistance from the master authority.

Indeed, as observed by [40], this is inherent for full deniability even in the simple case of IBE, due to an impossibility result of [10]. (Namely, while the result of [10] shows that fully receiver-deniable PKE is impossible, the result easily extends to IBE.) In the case that one of these parameters is polynomially unbounded, meaning that a bound is not fixed *a priori*, we denote it by

poly. Specifically, we envisage that the coercer should not be able to distinguish the equivocated coins of both the sender and receiver from an "honest transcript," even after requesting arbitrary secret keys and equivocations of its choice.

"Full" Receiver Deniability from Trapdoor Circuits. Next we show show how to transform any "IND-secure" FE scheme for general circuits (*i.e.*, where its functionality F computes general boolean circuits on some input length) into a FE for the same functionality that is (n_c, poly)-receiver-deniable in the full model (but where the receiver gets assistance from the master authority to equivocate) without introducing any additional assumption. In particular, recent works [26, 17, 44, 28] show IND-secure FE for general circuits whose security is based either on indistinguishable obfuscation and its variants or polynomial hardness of simple assumptions on multi-linear maps. We can use any of these schemes in our construction. We present a direct black-box transformation, making use of the "trapdoor circuits" technique, introduced by De Caro *et al.* [24] to show how to bootstrap IND-secure for circuits to the stronger notion of simulation security (SIM-security). The idea of the trapdoor mechanism is to replace the original circuit C with a trapdoor circuit $\mathsf{Trap}[C]$ that the *receiver faking algorithm* can then use to program the output in some way.

To give some intuition, let us consider for simplicity the case of equivocating a single ciphertext and secret key. Then, a plaintext will have two slots where the first slot will be the actual message x. The second slot will be a random string s, some sort of *tag* used to identify the ciphertext. On the other hand, $\mathsf{Trap}[C]$, where for simplicity we restrict C to be one-bit output circuit, will have two slots embedded in it, let us call them trapdoor values. Both the slots will be random strings r_1, r_2 used as formal variables to represent Boolean values 0 and 1. Now, if it happens that $s = r_1$ then $\mathsf{Trap}[C]$ returns 0, if $s = r_2$ then it returns 1, otherwise $\mathsf{Trap}[C]$ returns $C(x)$. Notice that, when s, r_1 and r_2 are chosen uniformly and independently at the random then the above events happen with negligible probability thus this trapdoor mechanism does not influence the correctness of the scheme. On the other hand, it is easy to see how the receiver faking algorithm works by setting r_1 or r_2 to s depending on the expected faked output. Clearly, the receiver needs the master authority to generate a new secret key, corresponding to circuit $\mathsf{Trap}[C]$, with tailored embedded r_1 and r_2. Moreover, the above solution fails when more secret keys have to be equivocated. In fact, s then would appear in all the faked secret keys and this would be easily recognizable by the the adversary. A trivial fix is to put in the ciphertexts as many different s's as the number of secret keys to be faked but this will create an unnecessary dependence that can be removed by using a PRF as a compact source of randomness. In the conference version [21] we presented the result in full details.

Efficient Receiver Deniable FE for Boolean Formulae. We study the possibility of achieving receiver deniability for weaker classes of functionalities more efficiently and assuming more standard assumptions. We show how to do this for Boolean formulae, namely we show how to transform any IND-secure FE scheme for Boolean formulate into one that is (n_c, n_d)-receiver deniable. Note that Katz, Sahai and Waters [35] show how to construct an FE scheme for Boolean formulae given an FE scheme for the inner-product predicate whose security, by the result of Okamoto and Takashima [38], can be based only on the Decisional Linear Assumption in bilinear groups. An interesting point, however, is that these schemes for boolean formulae allow polynomials in t variables with degree at most d in each variable, only if d^t is polynomial in the security parameter. This will mean that in order for our scheme to be efficient, the trapdoor mechanism will have a non-negligible probability of being activated by an honest encryption (i.e., correctness is non-negligible). We fix this issue by using parallel repetition. The resulting scheme achieves (n_c, n_k)-receiver deniable but we do not know how to achieve $n_k = \mathsf{poly}$.

"Multi-distributional" Receiver Deniability from "Delayed Trapdoor Circuits". In the previous receiver deniable FE schemes, the receiver crucially relies on the assistance of the master authority to generate a new secret key with tailored embedded r_1 and r_2, the trapdoor values. To avoid this, we need to find a way for the master authority to release a fake key that allows the receiver to modify the trapdoor values at any stage when required. This is solved using the new technique of *delayed trapdoor circuits*. Instead of embedding directly the trapdoor values in the $\mathsf{Trap}[C]$, they are encrypted using an CCA-like encryption scheme to avoid that the adversary can maul those values and learn something it should not learn. The resulting ciphertext, let us call it Ct', has to be linked to the corresponding $\mathsf{Trap}[C]$ by using a one-way function f as follows: a fresh random value z in the domain of f will be encrypted along with the trapdoor values, $t = f(z)$ will be embedded in trapdoor circuit. $\mathsf{Trap}[C]$ then will take in input also Ct' and verify that it encrypts a pre-image of t, before proceeding more. It is easy to see then that the fake key we were looking for is z. Knowing z allows to generate a new Ct' for different trapdoor values. Our construction starts from that of Garg *et al.* [26] but departs from it in many technicalities needed to face the challenges met in the hybrid experiments. For instance, as in Garg *et al.* a ciphertext of the functional encryption scheme for x corresponds to a double encryption, à la Naor-Yung [37], of x, using a statistical simulation-soundness NIZK. A secret key for circuit C is instead the differing-input obfuscation [6, 3, 17] of a trapdoor circuit $\mathsf{Trap}[C]$ that takes in input the double encryption of x and the double encryption of the trapdoor values related to $\mathsf{Trap}[C]$. Intuitively, differing-input obfuscation is required because there are certain Ct' that allows to discriminate, for example, which secret key $\mathsf{Trap}[C]$ is used to decrypt the double encryption of x. The actual construction is more involved, and is presented in full details in Section 4. We point out that in some concurrent works Apon *et al.*[4, 5] construct bi-deniable inner-product and attribute-based encryption schemes in the multidistributional model from LWE.

Sender deniability for FE from indistinguishability obfuscation. Sender deniability can be argued to be less important than receiver deniability in general. In fact the sender can always claim to have erased the random coins. Nonetheless, for some applications, sender deniability is fundamental. For instance, in some protocols the sender might have economic interest in selling the random coins (e.g., like it is the case in e-voting) and sender deniability is the right tool to prevent such problem.

We next show that any FE scheme for a functionality F can be upgraded to one that is $(\mathsf{poly}, \mathsf{poly})$-sender deniable assuming indistinguishability obfuscation. For this transformation we use the techniques introduced by Sahai and Waters in [41], who showed how to make any public-key encryption scheme sender-deniable. In this sense, we generalize their result from public-key encryption to any FE scheme for an arbitrary functionality. As compared to their result, we simply need to observe that their security proof holds even if the adversary knows the secret key of the starting encryption scheme. This result is presented in the Section 5 and the relevant definitions in Section 2.2.

We note that our assumption of indistinguishability obfuscation is already known to imply FE for the general circuit functionality [44], but we prefer to phrase our result as upgrading an arbitrary functionality rather than the former. The reason is that the techniques of Sahai and Waters [41] may not necessarily be tied to indistinguishability obfuscation, and if future work develops techniques for constructing deniable PKE based on weaker assumptions we hope they will transfer to our context as well.

We also note that this transformation also preserves receiver deniability of the starting scheme, and thus can be applied to the schemes obtained from our first result (on receiver deniability). In

this way we achieve (n_c, poly) bideniable FE for circuits assuming IND-secure FE for circuits and indistinguishability obfuscation. We remark that in the works of Apon *et al.*[4, 5] the notion of sender deniability achieved is multi-distributional.

Relation to Simulation-Based Security. As observed by [40], in the case of PKE, deniability implies a scheme is also *non-committing* [20, 22] and *secure under key-revealing selective-opening attacks* (SOA-K) [25, 8]. On the other hand, it was recently observed by [9] that the notion of simulation-based (SIM) security for FE implicitly incorporates SOA-K. SIM-security is a stronger notion of security for FE than IND-security and has been the subject of multiple recent works [39, 15, 9, 1, 24, 31, 7]. In both notions the adversary makes secret key queries interleaved with queries for challenge ciphertexts. IND-security asks that the adversary cannot distinguish between encryptions of messages that it cannot trivially distinguish using the keys. SIM-security requires that the "view" of the adversary can be simulated by a simulator given only the corresponding outputs of the functionality on the underlying plaintexts. This leads to the interesting result that a receiver-deniable FE scheme necessarily achieves some form of SIM-security. To formalize it, recall from [24] that (q_1, ℓ, q_2)-SIM security denotes SIM-security where the adversary is allowed to make at most q_1 non-adaptive key queries, ℓ encryption queries (challenge ciphertexts), and q_2 adaptive key queries. We show that an (n_c, n_k)-receiver deniable FE scheme is also $(0, n_c, n_k)$-SIM-secure (see the conference version [21] for a formal theorem and proof). On the other hand we stress deniability is *stronger* in the respect that equivocable ciphertexts and keys must decrypt correctly in the real system. Our results on receiver deniability can be seen as showing that the techniques of [24] are sufficient not just for achieving SIM-security but for deniability as well. Moreover, this implication implies that known impossibility results for SIM-secure FE [15, 9, 1, 24, 23, 2] mean that in the receiver deniable case (n_c, poly)-deniability (which we achieve assuming IND-secure FE for the circuit functionality) is in fact *optimal*. Iovino and Żebrowski [34] show how to overcome known impossibility results in the programmable RO model assuming that the time of decryption grow as the number of oracle queries. In the case of deniability it is unclear how programmable ROs could help since programmability only helps in a simulation whereas deniability refers to the behaviour of the real system. In other words, we show that the techniques of [24] are sufficient not only to achieve $(q_1, \ell, \mathsf{poly})$-SIM secure FE in the standard model but also receiver deniability for the analogous parameters (to add sender deniability we use different techniques, as SIM-security seems orthogonal).

2. Definitions

We first present formal definition of *functional encryption* [15, 27], and its security, and *deniable functional encryption* and its security. Due to space constraints we defer to [6, 3, 17] for definitions of differing-inputs obfuscation, and to Garg *et al.*. [26] for the definition of statistical simulation-sound non-interactive zero-knowledge proofs (SSS-NIZK, in short).

Functional Encryption. We define the primitive and its security following Boneh *et al.* [15] notation.

Definition 2.1. [Functionality] A *functionality* $F = \{F_n\}_{n>0}$ is a family of functions $F_n : K_n \times X_n \to \Sigma$ where K_n is the *key space* for parameter n, X_n is the *message space* for parameter n and Σ is the *output space*. We will refer to functionality F as a function from $F : K \times X \to \Sigma$ with $K = \cup_n K_n$ and $X = \cup_n X_n$.

Notice that, when $\vec{x} = (x_1, \ldots, x_\ell)$ is a vector of messages, for any $k \in K$, we denote by

$F(k, \vec{x})$ the vector of evaluations $(F(k, x_1), \ldots, F(k, x_\ell))$.

Definition 2.2. [Functional Encryption Scheme] A *functional encryption* scheme for functionality F defined over (K, X) is a tuple $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ of 4 algorithms with the following syntax:

1. $\mathsf{Setup}(1^\lambda, 1^n)$ outputs *public* and *master secret* keys $(\mathsf{Mpk}, \mathsf{Msk})$ for *security parameter* λ and *length parameter* n that are polynomially related.
2. $\mathsf{KeyGen}(\mathsf{Msk}, k)$, on input Msk for security parameter λ and $k \in K_n$ outputs *secret key* Sk.
3. $\mathsf{Enc}(\mathsf{Mpk}, x)$, on input Mpk for security parameter λ and $x \in X_n$ outputs *ciphertext* Ct;
4. $\mathsf{Dec}(\mathsf{Mpk}, \mathsf{Ct}, \mathsf{Sk})$ outputs $y \in \Sigma \cup \{\perp\}$.

In addition we make the following *correctness* requirement: for all $(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, all $k \in K_n$ and $x \in X_n$, the following probability taken over the random coins of the KeyGen and Enc algorithms is overwhelming: $\mathsf{Dec}(\mathsf{Mpk}, \mathsf{Ct}, \mathsf{Sk}) = F(k, x)$ whenever $F(k, x) \neq \perp$,[1] where $\mathsf{Sk} \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k)$ and $\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Mpk}, x)$.

Definition 2.3. [$\mathsf{Circuit}$ Functionality] The $\mathsf{Circuit}$ functionality has key space K_n equals to the set of all n-input Boolean circuits and message space X_n the set $\{0,1\}^n$ of n-bit strings. For $C \in K_n$ and $x \in X_n$, we have $\mathsf{Circuit}(C, x) = C(x)$, that is, the output of circuit C on input x.

Indistinguishability-based Security. The indistinguishability-based notion of security for functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for functionality F defined over (K, X) is formalized by means of the following game $\mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}$ between an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and a *challenger* $\mathcal{C}$.

$\mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda)$

1. $\mathcal{C}$ generates $(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and runs $\mathcal{A}_0$ on input Mpk;
2. $\mathcal{A}_0$, during its computation, issues q_1 *non-adaptive key-generation queries*. $\mathcal{C}$ on input key $k \in K$ computes $\mathsf{Sk} = \mathsf{KeyGen}(\mathsf{Msk}, k)$ and sends it to $\mathcal{A}_0$.
   When $\mathcal{A}_0$ stops, it outputs two *challenge messages vectors*, of length ℓ, $\vec{x}_0, \vec{x}_1 \in X^\ell$ and its internal state $\mathtt{st}$.
3. $\mathcal{C}$ picks $b \in \{0,1\}$ at random, and, for $i \in \ell$, computes the *challenge ciphertexts* $\mathsf{Ct}_i = \mathsf{Enc}(\mathsf{Mpk}, x_b[i])$. Then $\mathcal{C}$ sends $(\mathsf{Ct}_i)_{i \in [\ell]}$ to $\mathcal{A}_1$ that resumes its computation from state $\mathtt{st}$.
4. $\mathcal{A}_1$, during its computation, issues q_2 *adaptive key-generation queries*. $\mathcal{C}$ on input key $k \in K$ computes $\mathsf{Sk} = \mathsf{KeyGen}(\mathsf{Msk}, k)$ and sends it to $\mathcal{A}_1$.
5. When $\mathcal{A}_1$ stops, it outputs b'.
6. **Output:** if $b = b'$, for each $i \in [\ell]$, $|x_0^i| = |x_1^i|$, and $F(k, \vec{x}_0) = F(k, \vec{x}_1)$ for each k for which $\mathcal{A}$ has issued a key-generation query, then output 1 else output 0.

The advantage of adversary $\mathcal{A}$ in the above game is defined as

$$\mathsf{Adv}^{\mathsf{FE},\mathsf{IND}}_{\mathcal{A}}(1^\lambda) = \mathrm{Prob}[\mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda) = 1] - 1/2$$

Definition 2.4. We say that FE is (q_1, ℓ, q_2)-*indistinguishably secure* ((q_1, ℓ, q_2)-IND-Secure, for short) where $q_1 = q_1(\lambda), \ell = \ell(\lambda) q_2 = q_2(\lambda)$ are polynomials in the security parameter λ that are fixed a priori, if all probabilistic polynomial-time adversaries $\mathcal{A}$ issuing at most q_1 non-adaptive key queries, q_2 adaptive key queries and output challenge message vectors of length and most ℓ, have at most negligible advantage in the above game. Notice that, in the case that a parameter is an unbounded polynomial we use the notation poly. If a parameter is not specified then it assumed to be poly.

[1] See [9] for a discussion about this condition.

Receiver-Deniable Functional Encryption Scheme. We define the primitive and its security in the following way:

Definition 2.5. [Receiver-Deniable Functional Encryption Scheme] A (n_c, n_k)- *receiver-deniable functional encryption scheme for functionality F defined over (K, X)*, where $n_c = n_c(\lambda), n_k = n_k(\lambda)$ are polynomials in the security parameter λ that are fixed a priori, is made up of the algorithms $\mathsf{RecDenFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ of a standard FE scheme for F (Definition 2.2) and in addition the following algorithm:

- $\mathsf{RecFake}(\mathsf{Msk}, k, \mathbf{Ct}, \mathbf{x})$. The *receiver faking algorithm*, on input the master secret key Msk, a key k, at most n_c ciphertexts $\mathbf{Ct} = (\mathsf{Ct}_1, \ldots, \mathsf{Ct}_{n_c})$ and messages $\mathbf{x} = (x_1, \ldots, x_{n_c})$, outputs faked secret key Sk_C.

Correctness is defined as in Definition 2.2 and indistinguishability as in Definition 2.4.

Definition 2.6. [Receiver-Deniability] We require that for every PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, issuing at most n_k receiver-coerce queries to the oracles $\mathcal{K}_1$ and $\mathcal{K}_2$, the following two experiments are computationally indistinguishable.

$\mathsf{RealRecDenExp}^{\mathsf{RecDenFE}}_{\mathcal{A}}(1^\lambda)$	$\mathsf{FakeRecDenExp}^{\mathsf{RecDenFE}}_{\mathcal{A}}(1^\lambda)$
$(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$;	$(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$;
$(\mathbf{x}^\star, \mathbf{y}^\star, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathcal{O}_1, \mathcal{O}_2}(\mathsf{Mpk})$;	$(\mathbf{x}^\star, \mathbf{y}^\star, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathcal{O}_1, \mathcal{O}_2}(\mathsf{Mpk})$;
$(\mathsf{Ct}_i^\star \leftarrow \mathsf{Enc}(\mathsf{Mpk}, x_i; r_i))_{i \in [n_c]}$;	$(\mathsf{Ct}_i^\star \leftarrow \mathsf{Enc}(\mathsf{Mpk}, y_i; r_i))_{i \in [n_c]}$;
Output: $\mathcal{A}_1^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{K}_1(\cdot, \mathbf{Ct}^\star, \mathbf{x}^\star)}(\mathbf{Ct}^\star, \mathtt{st})$	**Output:** $\mathcal{A}_1^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{K}_2(\cdot, \mathbf{Ct}^\star, \mathbf{x}^\star)}(\mathbf{Ct}^\star, \mathtt{st})$

where $\mathbf{x}^\star = (x_1^\star, \ldots, x_{n_c}^\star)$, $\mathbf{y}^\star = (y_1^\star, \ldots, y_{n_c}^\star)$, and $\mathbf{Ct}^\star = (\mathsf{Ct}_1^\star, \ldots, \mathsf{Ct}_{n_c}^\star)$. $(\mathcal{K}_1, \mathcal{K}_2)$ are the receiver-coerce oracles.

All the oracles declared above are defined as follows:

$\mathcal{K}_1(k, \mathbf{Ct}, \mathbf{x})$	$\mathcal{K}_2(k, \mathbf{Ct}, \mathbf{x})$
$\mathsf{Sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k)$;	$\mathsf{Sk}_k \leftarrow \mathsf{RecFake}(\mathsf{Msk}, k, \mathbf{Ct}, \vec{x})$;
Output: Sk_k	**Output:** Sk_k

$\mathcal{O}_1(k, x, y)$	$\mathcal{O}_2(k, x, y)$
$\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Mpk}, x; r)$;	$\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Mpk}, y; r)$;
$\mathsf{Sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k)$;	$\mathsf{Sk}_k \leftarrow \mathsf{RecFake}(\mathsf{Msk}, k, \mathsf{Ct}, x)$;
Output: $(\mathsf{Ct}, \mathsf{Sk}_k)$	**Output:** $(\mathsf{Ct}, \mathsf{Sk}_k)$

In the above experiments, we require the following:

1. There is no query (k, x, y) issued to $\mathcal{O}_1$ and at same time a query $(k, \mathbf{Ct}^\star, \mathbf{x})$ for some $\mathbf{x}$ issued to $\mathcal{K}_1$ and there is no query (k, x, y) issued to $\mathcal{O}_2$ and at same time a query $(k, \mathbf{Ct}^\star, \mathbf{x})$ for some $\mathbf{x}$ issued to $\mathcal{K}_2$, where we consider all queries issued during the entire course of the experiment; i.e., when counting all the queries made by $\mathcal{A}_0$ and $\mathcal{A}_1$ together.
2. For any query issued by $\mathcal{A}_1$ to its oracle $\mathcal{K}_1$ or $\mathcal{K}_2$ oracle for key $k^\star$, neither $\mathcal{A}_0$ nor $\mathcal{A}_1$ queries $k^\star$ to either of their oracles $\mathcal{O}_1, \mathcal{O}_2$; i.e., they do not make any query $(k^\star, x, y)$ for some x, y to $\mathcal{O}_1$ or $\mathcal{O}_2$.
3. For each key k different from any of the challenge keys $k_i^\star$ queried by $\mathcal{A}_0$ and $\mathcal{A}_1$ to oracles $\mathcal{O}_1$ or $\mathcal{O}_2$, it holds that $F(k, \mathbf{x}^\star) = F(k, \mathbf{y}^\star)$.

2.1. Multi-Distributional Receiver-Deniable Functional Encryption Scheme

Definition 2.7. [Multi-Distributional Receiver-Deniable FE] A (n_c, n_k)-multi-distributional receiver-deniable functional encryption scheme for functionality F defined over (K, X), where $n_c = n_c(\lambda), n_k = n_k(\lambda)$ are polynomials in the security parameter λ that are fixed a priori, is made up of the algorithms $\mathsf{MDRecDenFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ of a standard FE scheme for F (Definition 2.2) and in addition the following two algorithms:

- $\mathsf{DenKeyGen}(\mathsf{Msk}, k)$. The *deniable key generation algorithm*, on input the master secret key Msk, and key k, outputs secret key Sk_k and fake key Fk_k.
- $\mathsf{RecFake}(\mathsf{Sk}_k, \mathsf{Fk}_k, \mathbf{Ct}, \mathbf{x})$. The *receiver faking algorithm*, on input secret key and fake key $\mathsf{Sk}_k, \mathsf{Fk}_k$ for key k, at most n_c ciphertexts $\mathbf{Ct} = (\mathsf{Ct}_1, \ldots, \mathsf{Ct}_{n_c})$ and messages $\mathbf{x} = (x_1, \ldots, x_{n_c})$, outputs faked secret key Sk'_k.

Correctness is defined as in Definition 2.2 and indistinguishability as in Definition 2.4. We also require the following security property.

Definition 2.8. [Multi-Distributional Receiver Deniability] We require that for every PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, issuing at most n_k receiver-coerce queries to the oracles $\mathcal{K}_1$ and $\mathcal{K}_2$, the following two experiments are computationally indistinguishable.

$\mathsf{RealMDRecDenExp}^{\mathsf{RecDenFE}}_{\mathcal{A}}(1^\lambda)$	$\mathsf{FakeMDRecDenExp}^{\mathsf{RecDenFE}}_{\mathcal{A}}(1^\lambda)$
$(\mathbf{x}^\star, \mathbf{y}^\star, \mathtt{st}) \leftarrow \mathcal{A}_0(1^\lambda)$;	$(\mathbf{x}^\star, \mathbf{y}^\star, \mathtt{st}) \leftarrow \mathcal{A}_0(1^\lambda)$;
$(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$;	$(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$;
$(\mathsf{Ct}^\star_i \leftarrow \mathsf{Enc}(\mathsf{Mpk}, x_i; r_i))_{i\in[n_c]}$;	$(\mathsf{Ct}^\star_i \leftarrow \mathsf{Enc}(\mathsf{Mpk}, y_i; r_i))_{i\in[n_c]}$;
Output: $\mathcal{A}_1^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{K}_1(\cdot, \mathbf{Ct}^\star, \mathbf{x}^\star)}(\mathsf{Mpk}, \mathbf{Ct}^\star, \mathtt{st})$	**Output:** $\mathcal{A}_1^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{K}_2(\cdot, \mathbf{Ct}^\star, \mathbf{x}^\star)}(\mathsf{Mpk}, \mathbf{Ct}^\star, \mathtt{st})$

where $\mathbf{x}^\star = (x^\star_1, \ldots, x^\star_{n_c})$, $\mathbf{y}^\star = (y^\star_1, \ldots, y^\star_{n_c})$, and $\mathbf{Ct}^\star = (\mathsf{Ct}^\star_1, \ldots, \mathsf{Ct}^\star_{n_c})$. $(\mathcal{K}_1, \mathcal{K}_2)$ are the receiver-coerce oracles.

All the oracles declared above are defined as follows:

$\mathcal{K}_1(k, \mathbf{Ct}, \mathbf{x})$	$\mathcal{K}_2(k, \mathbf{Ct}, \mathbf{x})$
$\mathsf{Sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k)$;	$(\mathsf{Sk}_k, \mathsf{Fk}_k) \leftarrow \mathsf{DenKeyGen}(\mathsf{Msk}, k)$;
Output: Sk_k	$\mathsf{Sk}'_k \leftarrow \mathsf{RecFake}(\mathsf{Sk}_k, \mathsf{Fk}_k, \mathbf{Ct}, \mathbf{x})$;
	Output: Sk'_k

$\mathcal{O}_1(k, x, y)$	$\mathcal{O}_2(k, x, y)$
$\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Mpk}, x; r)$;	$\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Mpk}, y; r)$;
$\mathsf{Sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k)$;	$(\mathsf{Sk}_k, \mathsf{Fk}_k) \leftarrow \mathsf{DenKeyGen}(\mathsf{Msk}, k)$;
Output: $(\mathsf{Ct}, \mathsf{Sk}_k)$	$\mathsf{Sk}'_k \leftarrow \mathsf{RecFake}(\mathsf{Sk}_k, \mathsf{Fk}_k, \mathsf{Ct}, x)$;
	Output: $(\mathsf{Ct}, \mathsf{Sk}'_k)$

In the above experiments, we require the following:

1. There is no query (k, x, y) issued to $\mathcal{O}_1$ and at same time a query $(k, \mathbf{Ct}^\star, \mathbf{x})$ for some $\mathbf{x}$ issued to $\mathcal{K}_1$ and there is no query (k, x, y) issued to $\mathcal{O}_2$ and at same time a query $(k, \mathbf{Ct}^\star, \mathbf{x})$ for some $\mathbf{x}$ issued to $\mathcal{K}_2$, where we consider all queries issued during the entire course of the experiment; i.e., when counting all the queries made by $\mathcal{A}_0$ and $\mathcal{A}_1$ together.
2. For any query issued by $\mathcal{A}_1$ to its oracle $\mathcal{K}_1$ or $\mathcal{K}_2$ for key $k^\star$, neither $\mathcal{A}_0$ nor $\mathcal{A}_1$ queries $k^\star$ to either of their oracles $\mathcal{O}_1, \mathcal{O}_2$; i.e., they do not make any query $(k^\star, x, y)$ for some x, y to $\mathcal{O}_1$ or $\mathcal{O}_2$.

3. For each key k different from any of the challenge keys $k_i^\star$ queried by $\mathcal{A}$ to oracles $\mathcal{O}_1$ or $\mathcal{O}_2$, it holds that $F(k, \mathbf{x}^\star) = F(k, \mathbf{y}^\star)$.

Remark 2.9. *Our security notion is* selective*, in that the adversary* commits *to* (x, y) *before it sees* Mpk*. It is possible to bootstrap selectively-secure scheme to full security using standard complexity leveraging arguments [12, 32] at the price of a* $2^{|x|}$ *loss in the security reduction.*

2.2. Sender-Deniable Functional Encryption Scheme

In this Section we define sender-deniable FE. First we recall some relevant definitions.

Publicly Deniable Encryption In [41], Sahai and Waters introduced the strong notion of *publicly deniable encryption* and the related security properties. Specifically:

Definition 2.10. [Publicly Deniable Encryption] A *publicly deniable encryption scheme* over a message space $X = X_\lambda$ consists of four algorithms $\mathsf{PDE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Explain})$ with the following semantics:

- $\mathsf{Setup}(1^\lambda)$ is a polynomial time algorithm that takes as input the unary representation of the security parameter λ and outputs a public key Mpk and a secret key Sk.
- $\mathsf{Enc}(\mathsf{Mpk}, x \in X; u)$ is a polynomial time algorithm that takes as input the public key Mpk a message x and uses random coins u. It outputs a ciphertext Ct.
- $\mathsf{Dec}(\mathsf{Sk}, \mathsf{Ct})$ is a polynomial time algorithm that takes as input a secret key Sk and ciphertext Ct, and outputs a message x.
- $\mathsf{Explain}(\mathsf{Mpk}, \mathsf{Ct}, x; r)$ is a polynomial time algorithm that takes as input a public key Mpk, a ciphertext Ct, and a message x, and outputs and a string e, that is the same size as the randomness u taken by Enc algorithm above.

We say that a publicly deniable encryption scheme is correct if for all messages $x \in X$

$$\Pr[(\mathsf{Mpk}, \mathsf{Sk}) \leftarrow \mathsf{Setup}(1^\lambda); \mathsf{Dec}(\mathsf{Sk}, \mathsf{Enc}(\mathsf{Mpk}, x; u)) \neq x] = \mathsf{negl}(\lambda)$$

The security requirements for a publicly deniable encryption system are two and are defined as follows.

Indistinguishability-based Security. This is exactly the same security game [30] as in standard public key encryption schemes.

Indistinguishability of Explanation. Formally, the indistinguishability of explanation notion of security for public deniable encryption scheme $\mathsf{PDE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Explain})$ is formalized by means of the following game $\mathsf{IND\text{-}EXPL}_{\mathcal{A}}^{\mathsf{PDE}}$ between an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and a *challenger* $\mathcal{C}$.

$\mathsf{IND\text{-}EXPL}_{\mathcal{A}}^{\mathsf{PDE}}(1^\lambda)$

1. $\mathcal{C}$ generates $(\mathsf{Mpk}, \mathsf{Sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and runs $\mathcal{A}_0$ on input Mpk;
2. When $\mathcal{A}_0$ stops, it outputs a *single* message $x \in X$ and its internal state $\mathtt{st}$. $\mathcal{C}$ creates $\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Mpk}, x; u)$ for random u and it computes $e \leftarrow \mathsf{Explain}(\mathsf{Mpk}, \mathsf{Ct}, x; r)$ for random r. Then, $\mathcal{C}$ flips a coin $b \leftarrow \{0, 1\}$. If $b = 0$, $\mathcal{C}$ sends (Ct, u) to $\mathcal{A}_1$ that resumes its computation from state $\mathtt{st}$. Otherwise, if $b = 1$, $\mathcal{C}$ sends (Ct, e).
3. When $\mathcal{A}_1$ stops, it outputs b'.
4. **Output:** if $b = b'$, then output 1 else output 0.

The advantage of adversary $\mathcal{A}$ in the above game is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PDE},\mathsf{IND\text{-}EXPL}}(1^\lambda) = \mathrm{Prob}[\mathsf{IND\text{-}EXPL}_{\mathcal{A}}^{\mathsf{PDE}}(1^\lambda) = 1] - 1/2$$

Definition 2.11. We say that PDE has *indistinguishably of explanation* if all probabilistic polynomial time adversaries $\mathcal{A}$ have at most negligible advantage in the above game.

Generic Transformation. In [41], Sahai and Waters show how to transform any public key encryption scheme $\mathcal{E}$ in a publicly deniable encryption scheme PDE by using indistinguishability obfuscation and punctured pseudo-random functions [29, 16, 36, 18]. In particular, for the so obtained PDE scheme, indistinguishability of explanation holds also when the adversary is given the secret key of the underlying public key encryption scheme $\mathcal{E}$.

Publicly Deniable Functional Encryption

Definition 2.12. [Publicly Deniable Functional Encryption] A *publicly deniable functional encryption scheme* PDFE for functionality F defined over (K, X) consists of the algorithms that make up a functional encryption scheme for the same functionality (Definition 2.2), with in addition an explanation algorithm as defined in Definition 2.10.

Security.: Indistinguishability-based security and indistinguishability of explanation of PDFE are defined exactly the same way as Definition 2.4 and 2.11, respectively.

3. Receiver Deniable FE for Boolean Formulae

The previous construction for general circuits is based on obfuscation and thus represents currently only a feasibility result.

In this section we consider receiver deniability for weaker classes of functionalities that still support some form of trapdoor mechanism and for which a functional encryption scheme can be constructed more efficiently and assuming standard assumptions. Namely, we are interested in constructing a receiver deniable FE for Boolean formulae in which Boolean variables are encrypted and the receiver, having a secret key for a Boolean formula, test whether the variables in the ciphertext satisfy the formula in the key. In [35], Katz, Sahai and Waters show how to construct a functional encryption scheme for Boolean formulae given a functional encryption scheme for the inner-product whose security, by the result of Okamoto and Takashima [38], can be based only on the Decisional Linear Assumption in bilinear groups. To construct a functional encryption scheme for Boolean formulae, [35] first shows how to construct functional encryption schemes for predicates corresponding to univariate polynomials whose degree d is at most polynomial in the security parameter. This can be generalized to the case of polynomials in t variables, and degree at most d in each variable but the drawback is that d^t has to be polynomial in the security parameter. Given the polynomial encryption construction, [35] shows that for Boolean variables it is possible to handle CNF or DNF formulae by observing that the predicate OR_{I_1,I_2}, where $\mathsf{OR}_{I_1,I_2}(x_1, x_2) = 1$ iff either $x_1 = I_1$ or $x_2 = I_2$, can be encoded as the bivariate polynomial $p(x_1, x_2) = (x_1 - I_1) \cdot (x_2 - I_2)$ and the predicate AND_{I_1,I_2}, where $\mathsf{AND}_{I_1,I_2}(x_1, x_2) = 1$ if both $x_1 = I_1$ and $x_2 = I_2$, correspond to the polynomial $p(x_1, x_2) = (x_1 - I_1) + (x_2 - I_2)$. The complexity of the resulting scheme depends polynomially on d^t, where t is the number of variables and d is the maximum degree of the resulting polynomial in each variable. This bound will critically influence our construction of receiver deniable scheme as we will show in the next section. Specifically, the length of the additional slots used in trapdoor mechanism of the previous section will be independent of the security parameter to avoid the exponential blowup of the complexity of the resulting scheme. As a consequence, the trapdoor mechanism has unluckily a non-negligible probability of being active in the real scheme thus influencing the decryption error probability. Finally, parallel repetition will

fix this issue and the overall construction will have negligible error and efficiency polynomially related to the one of the underlying inner-product encrption scheme.

3.1. Our Construction

Overview.: The trapdoor formula will follow the same design of the trapdoor circuit construction with the main difference being the length of the slots which will be here constant and independent from the security parameter to avoid the exponential blowup in the [35]'s construction. The plaintext will have two slots where the first slot will be the actual message x to encrypt. The second slot will be a random string s. On the other hand, a secret key for Boolean formula f will also have two slots to represent Boolean values 0 and 1. Specifically:

Construction 3.1. [Trapdoor Boolean Formula] Let f be a Boolean formula on n-bits. For any two strings $r_0, r_1 \in \{0,1\}^\ell$, define the corresponding *trapdoor boolean formula* $\mathsf{Trap}[f]^{r_0,r_1}$ on $(n+\ell)$-bit inputs as follows: **Formula**$\mathsf{Trap}[f]^{r_0,r_1}(x,s) := (s = r_1) \vee [f(x) \wedge \neg(s = r_0)]$, where the expression $(s = r)$ is the comparison bit-a-bit.

We are now ready to present our RecDenFE scheme.

Construction 3.2. [Receiver Deniable Functional Encryption for Boolean Formulae] Let $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.KeyGen}, \mathsf{FE.Eval})$ be the functional encryption scheme for the functionality Boolean Formulae. For any constant $\ell > 3$, we define our *receiver deniable functional encryption scheme* $\mathsf{RecDenFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{RecFake})$ for Boolean formulae as follows.

- $\mathsf{Setup}(1^\lambda, 1^n, 1^m)$, for each $i \in [m]$, runs $\mathsf{FE.Setup}(1^\lambda, 1^{n+\ell})$ to get the pair $(\mathsf{FE.Mpk}_i, \mathsf{FE.Msk}_i)$. Then, the master public key is $\mathsf{Mpk} = (\mathsf{FE.Mpk})_{i\in[m]}$ and the master secret key is $\mathsf{Msk} = (\mathsf{FE.Msk})_{i\in[m]}$. The algorithm returns the pair $(\mathsf{Mpk}, \mathsf{Msk})$.
- $\mathsf{Enc}(\mathsf{Mpk}, x)$, on input master public key $\mathsf{Mpk} = (\mathsf{FE.Mpk}_i)_{i\in[m]}$ and message $x \in \{0,1\}^n$, for each $i \in [m]$, chooses a random $s_i \in \{0,1\}^\ell$ and sets $\mathsf{Ct}_i = \mathsf{FE.Enc}(\mathsf{FE.Mpk}_i, (x, s_i))$. The algorithm returns the ciphertext $\mathsf{Ct} = (\mathsf{Ct}_i)_{i\in[m]}$.
- $\mathsf{KeyGen}(\mathsf{Msk}, f)$, on input master secret key $\mathsf{Msk} = (\mathsf{FE.Msk})_{i\in[m]}$ and a n-input Boolean formula f, for each $i \in [m]$, chooses two random strings $r_0^i, r_1^i \in \{0,1\}^\ell$, such that $r_0^i \neq r_1^i$, and computes secret key $\mathsf{FE.Sk}_f^i = \mathsf{FE.KeyGen}(\mathsf{FE.Msk}_i, \mathsf{Trap}[f]^{r_0^i, r_1^i})$. The algorithm returns the secret key $\mathsf{Sk}_f = (r_0^i, r_1^i, \mathsf{FE.Sk}_f^i)_{i\in[m]}$.
- $\mathsf{Dec}(\mathsf{Mpk}, \mathsf{Ct}, \mathsf{Sk}_f)$, on input master public key $\mathsf{Mpk} = (\mathsf{FE.Mpk})_{i\in[m]}$, $\mathsf{Ct} = (\mathsf{Ct}_i)_{i\in[m]}$ and secret key $\mathsf{Sk}_f = (r_0^i, r_1^i, \mathsf{FE.Sk}_f^i)_{i\in[m]}$ for Boolean formula f, for $i \in [m]$, computes Boolean value $b_i = \mathsf{FE.Eval}(\mathsf{FE.Mpk}_i, \mathsf{Ct}_i, \mathsf{FE.Sk}_f^i)$, and returns as output the Boolean value on which the majority of b_i's have agreed on.
- $\mathsf{RecFake}(\mathsf{Msk}, f, \mathsf{Ct}, x')$, on input the master secret key $\mathsf{Msk} = (\mathsf{FE.Msk}_i)_{i\in[m]}$, an n-input Boolean formula f, ciphertext $\mathsf{Ct} = (\mathsf{Ct}_i)_{i\in[m]}$ and message x', for all $i \in [m]$, the algorithm extracts s_i from Ct_i by using $\mathsf{FE.Msk}_i$. Now, RecFake chooses the r_0^i's and r_1^i's by following a binomial distribution with number of trials equals to m and success probability $p = (1 - 2^{-\ell})$. Specifically, RecFake distinguishes between the following two cases. Let $b' = f(x')$, for each $i \in [m]$, if there is a success in the i-th trial then RecFake sets $r_{b'}^i = s_i$ and r_{1-b}^i to a random value different from s_i. Otherwise, RecFake sets $r_{1-b'}^i = s_i$ and r_b to a random value different from s_i. Finally, RecFake computes secret key $\mathsf{FE.Sk}_f^i = \mathsf{FE.KeyGen}(\mathsf{FE.Msk}_i, \mathsf{Trap}[f]^{r_0^i, r_1^i})$, and returns the secret key $\mathsf{Sk}_f = (r_0^i, r_1^i, \mathsf{FE.Sk}_f^i)_{i\in[m]}$ as faking key.

Correctness. Notice that for any $i \in [m]$, the probability that $s_i = r_0^i \vee s_i = r_1^i$ is at most $2^{-\ell+1}$.

Thus the output of the decryption is correct, i.e. $\mathsf{Trap}[f]^{r_0^i,r_1^i}(x,s_i) = f(x)$, with probability at least $1-2^{-\ell+1}$.

Thus on average, an $(1-2^{-\ell+1})$ fraction of the ciphertexts will be decrypted to the correct value and for large enough m, the Chernoff bound guarantees that the correctness of RecDenFE hold with overwhelming probability.

Security. The proof that RecDenFE is a $(1,1)$-receiver deniable functional encryption scheme for Boolean formulae is essentially that for general functionalities of the conference version [21] and we omit further details. In fact, observe that Boolean formulae are a very special case of general circuits and as such the proof follows the same structure.

We note that one can extend the scheme to (n_c, n_k)-receiver deniability in a simple way but we cannot achieve $n_k = \mathsf{poly}$ in this case, however, because we cannot use symmetric encryption (at least in a straightforward way).

4. Multi-Distributional Receiver Deniable FE from di$\mathcal{O}$

To not overburden the notation and to make the presentation easier to follow, we present a construction of a $(1,1)$-*multi-distributional receiver deniable functional encryption scheme for* Circuit.

4.0.1. Overview.: Our construction resembles that of Garg *et al.* [26] but with some differences. A ciphertext of the functional encryption scheme for x corresponds to a double encryption, à la Naor-Yung [37], of x, using a statistical simulation-soundness NIZK. Nevertheless, in our construction a secret key for circuit C is the differing-input obfuscation of a trapdoor circuit $\mathsf{Trap}[C]$ that takes in input the double encryption of x and the double encryption of the trapdoor values.

Construction 4.1. [Multi-Distributional Receiver Deniable FE] Given an IND-CPA PKE system $\mathcal{E} = (\mathcal{E}.\mathsf{Setup}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec})$ with perfect correctness, a differing-inputs obfuscator di$\mathcal{O}$, an SSS-NIZK proof system $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify}, \mathsf{NIZK.Sim})$ and a one-way function f, we define our multi-distributional receiver deniable functional encryption $\mathsf{MDRecDenFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{DenKeyGen}, \mathsf{RecFake}, \mathsf{Dec})$ as follows:

1. $\mathsf{Setup}(1^\lambda)$ takes in input the *security parameter* λ and computes the following: For $i \in [4]$, $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathcal{E}.\mathsf{Setup}(1^\lambda)$. Then, $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. The algorithm sets $\mathsf{Mpk} = ((\mathsf{pk}_i)_{i\in[4]}, f, \mathsf{crs})$, $\mathsf{Msk} = ((\mathsf{sk}_i)_{i\in[4]})$.
2. $\mathsf{KeyGen}(\mathsf{Msk}, C)$ takes in input *master secret key* $\mathsf{Msk} = ((\mathsf{sk}_i)_{i\in[4]})$, and *circuit* C, and does the following: Computes common reference string $\mathsf{crs}' \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. Then, sample random z in the domain of f and set $t = f(z)$ and compute $\mathsf{Ct}' := (\mathsf{ct}_3, \mathsf{ct}_4, \pi_2)$, where $\mathsf{ct}_3 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_3, (z, 0^n, 0^\lambda); r_3)$ and $\mathsf{ct}_4 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_4, (z, 0^n, 0^\lambda); r_4)$, and π_2 is a NIZK proof of Equation 2. Finally, the algorithm computes a differing-input obfuscation $\mathsf{di}\mathcal{O}_{\mathsf{Trap}_{1,3}}$ for the trapdoor circuit $\mathsf{Trap}_{1,3}[C, \mathsf{crs}, \mathsf{crs}', \mathsf{sk}_1, \mathsf{sk}_3, f, t]$. The algorithm outputs *secret key* for the circuit C, $\mathsf{Sk}_C = (\mathsf{di}\mathcal{O}_{\mathsf{Trap}_{1,3}}, t, \mathsf{Ct}')$.
3. $\mathsf{DenKeyGen}(\mathsf{Msk}, C)$ takes in input$\mathsf{Msk} = ((\mathsf{sk}_i)_{i\in[4]})$ and *circuit* C, and computes $\mathsf{Sk}_C = \mathsf{KeyGen}(\mathsf{Msk}, C)$. The algorithm outputs Sk_C as the *secret key* for the circuit C and $\mathsf{Fk}_C = z$.
4. $\mathsf{Enc}(\mathsf{Mpk}, x)$, on input *master public key* $\mathsf{Mpk} = ((\mathsf{pk}_i)_{i\in[4]}, f, \mathsf{crs})$ and *messages* $x \in X_n$, computes $\mathsf{Ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi_1)$, where $\mathsf{ct}_1 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_1, x; r_1)$ and $\mathsf{ct}_2 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_2, x; r_2)$ and π_1 is a NIZK proof of Equation 1. The algorithm outputs *ciphertext* Ct.
5. $\mathsf{Dec}(\mathsf{Sk}_C, \mathsf{Ct})$ on input *secret key* $\mathsf{Sk}_C = (\mathsf{di}\mathcal{O}_{\mathsf{Trap}_{1,3}}, t, \mathsf{Ct}')$ and *ciphertext* Ct, the algorithm outputs $\mathsf{di}\mathcal{O}_{\mathsf{Trap}_{1,3}}(\mathsf{Ct}, \mathsf{Ct}')$.

6. $\mathsf{RecFake}(\mathsf{Sk}_C, \mathsf{Fk}_C, \mathsf{Ct}, x)$ on input *secret key* $\mathsf{Sk}_C = (\mathsf{di}\mathcal{O}_{\mathsf{Trap}_{1,3}}, t, \mathsf{Ct}')$, *fake key* $\mathsf{Fk}_C = z$, where $t = f(z)$, *ciphertext* $\mathsf{Ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi_1)$ and *message* x, does the following: Compute $\hat{\mathsf{Ct}} := (\hat{\mathsf{ct}}_3, \hat{\mathsf{ct}}_4, \hat{\pi}_2)$, where $\hat{\mathsf{ct}}_3 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_3, (z, \mathsf{Ct}, x); r_3)$ and $\hat{\mathsf{ct}}_4 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_4, (z, \mathsf{Ct}, x); r_4)$ and $\hat{\pi}_2$ is a NIZK proof of Equation 2. The new secret key for circuit C is $\mathsf{Sk}_C = (\mathsf{di}\mathcal{O}_{\mathsf{Trap}_{1,3}}, t, \hat{\mathsf{Ct}})$.

Correctness follows immediately from the correctness of the di$\mathcal{O}$, PKE, SSS-NIZK, and the description of the trapdoor circuits described below.

$\mathsf{Trap}_{i,j}[C, \mathsf{crs}, \mathsf{crs}', \mathsf{sk}_i, \mathsf{sk}_j, f, t](\mathsf{Ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi_1), \mathsf{Ct}' = (\mathsf{ct}_3, \mathsf{ct}_4, \pi_2))$

The algorithm does the following:

1. Check that π_1 is valid NIZK proof (using the $\mathsf{NIZK.Verify}$ algorithm and crs) for the NP-statement

$$\begin{aligned} &\exists x, r_1, r_2 : \\ &\mathsf{ct}_1 = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_1, x; r_1) \text{ and } \mathsf{ct}_2 = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_2, x; r_2) \end{aligned} \tag{1}$$

2. Check that π_2 is valid NIZK proof (using the $\mathsf{NIZK.Verify}$ algorithm and crs') for the NP-statement

$$\begin{aligned} &\exists z, c, x, r_3, r_4 : \\ &\mathsf{ct}_3 = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_3, (z, c, x); r_3) \text{ and } \mathsf{ct}_4 = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_4, (z, c, x); r_4) \text{ and } f(z) = t \end{aligned} \tag{2}$$

3. If any check fails output 0.
4. $(z', c', x') \leftarrow \mathcal{E}.\mathsf{Dec}(\mathsf{sk}_j, \mathsf{ct}_j)$
5. if $c' = \mathsf{Ct}$ then output $C(x')$; otherwise output $C(\mathcal{E}.\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i))$.

Remark 4.2. *To allow a secret key to support faking against n_c ciphertexts, like it is done in the receiver deniable scheme of the conference version, we attach to a secret key n_c ciphertexts Ct'_i, each being the double encryption of $(z_i, 0^n, 0^\lambda)$ for different z_i's. Then, $\mathsf{Trap}_{i,j}$ will contain the images under the one-way function f of all z_i's.*

4.0.2. Proof of Security.:

We state the following theorem.

Theorem 4.3. If di$\mathcal{O}$ is an differing-input obfuscator, $\mathcal{E}$ is IND-CPA and f is a one-way function then MDRecDenFE is a $(1, 1)$-*multi-distributional receiver deniable* in the sense of Definition 2.6.

The proof is given in [21].

5. Sender Deniability via Indistinguishability Obfuscation

In this section, we show how to add sender deniability to any FE scheme for a functionality F by using the techniques introduced by Sahai and Waters in [41] for public deniability. Furthermore, the construction preserves any form of receiver deniability the starting scheme has. In particular, applying by adding sender deniability to the receiver deniable FE scheme of the conference version [21], we get a *bi-deniable FE scheme for* Circuit (DenFE, for short),

In Section 2.2 we introduced the *publicly deniable functional encryption* primitive and we now show how to modify the receiver deniable scheme of the conference version [21] to add sender deniability.

Construction.: By applying the Sahai-Waters [41] generic transformation to a functional encryption FE, we obtain a publicly deniable functional encryption scheme PDFE whose indistinguishability of explanation holds also when *the adversary is given the master secret key of the*

underlying functional encryption scheme FE. We will use critically this property during the security proof of our DenFE scheme. In fact, when invoking the indistinguishability of explanation having the master secret key of the underlying functional encryption scheme enables the simulator to generate the secret keys requested by the adversary.

5.1. Our Construction

Overview. The main difference with the receiver deniable FE of the conference version [21] is that we now employ a publicly deniable functional encryption scheme. Then the *sender faking algorithm* will correspond to the PDFE's explanation algorithm. Specifically, our DenFE scheme is defined as follows:

Definition 5.1. [Bideniable Functional Encryption for Circuit] Let $\mathsf{PDFE} = (\mathsf{PDFE.Setup}, \mathsf{PDFE.Enc}, \mathsf{PDFE.KeyGen}, \mathsf{PDFE.Eval}, \mathsf{PDFE.Explain})$ be the publicly deniable functional encryption scheme for the functionality Circuit. and $\mathcal{F} = \{f_s : s \in \{0,1\}^\lambda\}_{\lambda\in\mathbb{N}}$ be a pseudo-random function family of functions with domain $\{0,1\}^{l(\lambda)}$ and range $\{0,1\}^{L(\lambda)}$. We define our *bi-deniable functional encryption scheme* $\mathsf{DenFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{SendFake}, \mathsf{RecFake})$ for Circuit as follows.

- $\mathsf{Setup}(1^\lambda, 1^n)$ runs $\mathsf{PDFE.Setup}(1^\lambda, 1^{n+\lambda})$ to get the pair $(\mathsf{PDFE.Mpk}, \mathsf{PDFE.Msk})$. Then, the master public key is $\mathsf{Mpk} = \mathsf{PDFE.Mpk}$ and the master secret key is $\mathsf{Msk} = \mathsf{PDFE.Msk}$. The algorithm returns the pair $(\mathsf{Mpk}, \mathsf{Msk})$.
- $\mathsf{Enc}(\mathsf{Mpk}, x)$ on input master public key $\mathsf{Mpk} = \mathsf{PDFE.Mpk}$, and message $x \in \{0,1\}^n$, chooses a random $s \in \{0,1\}^\lambda$ and sets $x' = (x, s)$. Then the algorithm computes and returns the ciphertext

$$\mathsf{Ct} = \mathsf{PDFE.Enc}(\mathsf{PDFE.Mpk}, x') \ .$$

- $\mathsf{KeyGen}(\mathsf{Msk}, C)$ on input master secret key $\mathsf{Msk} = \mathsf{PDFE.Msk}$ and a n-input Boolean circuit C, chooses, for $i \in [n_c]$, random strings $t_i, t'_i \in \{0,1\}^{l(\lambda)}$, $z_i, z'_i \in \{0,1\}^{L(\lambda)}$ and computes

$$\mathsf{PDFE.Sk}_C = \mathsf{PDFE.KeyGen}(\mathsf{PDFE.Msk}, \mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'}) \ ,$$

  where $\mathbf{t} = (t_1, \ldots, t_{n_c}), \mathbf{z} = (z_1, \ldots, z_{n_c}), \mathbf{t}' = (t'_1, \ldots, t'_{n_c}), \mathbf{z}' = (z'_1, \ldots, z'_{n_c})$ and $\mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'}(x, s)$ is defined in the following way: if there exists some $i \in [n_c]$ such that $f_s(t_i) = z_i$ (resp. $f_s(t'_i) = z'_i$), then return 1 (resp. 0), otherwise return $C(x)$.

  The algorithm returns the secret key $\mathsf{Sk}_C = (\mathbf{t}, \mathbf{z}, \mathbf{t}', \mathbf{z}', \mathsf{PDFE.Sk}_C)$.
- $\mathsf{Eval}(\mathsf{Mpk}, \mathsf{Ct}, \mathsf{Sk}_C)$ on input master public key $\mathsf{Mpk} = \mathsf{PDFE.Mpk}$, Ct and secret key $\mathsf{Sk}_C = (\mathbf{t}, \mathbf{z}, \mathbf{t}', \mathbf{z}', \mathsf{PDFE.Sk}_C)$ for circuit C, returns the output of $\mathsf{PDFE.Eval}(\mathsf{PDFE.Mpk}, \mathsf{Ct}, \mathsf{PDFE.Sk}_C)$.
- $\mathsf{SendFake}(\mathsf{Mpk}, \mathsf{Ct}, x)$ on input master public key $\mathsf{Mpk} = \mathsf{PDFE.Msk}$, ciphertext Ct and message $x \in \{0,1\}^n$, chooses a random $s \in \{0,1\}^\lambda$ and sets $x' = (x, s)$. Then, SendFake runs PDFE.Explain on input ciphertext Ct and message x' to get faked randomness e, and returns $r_S = s||e$ as faked sender randomness.
- $\mathsf{RecFake}(\mathsf{Msk}, C, \mathbf{Ct}, \mathbf{x})$ on input the master secret key $\mathsf{Msk} = \mathsf{PDFE.Msk}$, a Boolean circuit C on n-bits input and 1-bit output, at most n_c ciphertexts $\mathbf{Ct} = (\mathsf{Ct}_1, \ldots, \mathsf{Ct}_\ell)$, for $\ell \leq n_c$, and messages $\mathbf{x} = (x_1, \ldots, x_\ell)$, extracts s_i from each ciphertext Ct_i by using $\mathsf{PDFE.Msk}$. Then, for each $i \in [\ell]$, RecFake chooses random t_i and t'_i in $\{0,1\}^{l(\lambda)}$ and distinguishes between the following two case:
  - If $C(x_i) = 1$, it sets $z_i = f_{s_i}(t_i)$ and chooses random $z'_i \in \{0,1\}^{L(\lambda)}$.
  - If $C(x_i) = 0$, it sets $z'_i = f_{s_i}(t'_i)$ and chooses random $z_i \in \{0,1\}^{L(\lambda)}$.

  Finally, RecFake computes $\mathsf{PDFE.Sk}_C = \mathsf{PDFE.KeyGen}(\mathsf{PDFE.Msk}, \mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'})$, and returns secret key $\mathsf{Sk}_C = (\mathbf{t}, \mathbf{z}, \mathbf{t}', \mathbf{z}', \mathsf{PDFE.Sk}_C)$.

Correctness of our DenFE scheme follows from the correctness of the PDFE scheme and from the observation that, for randomly chosen $\mathbf{t}, \mathbf{z}, \mathbf{t}', \mathbf{z}'$ and s and for all x, $\mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'}(x, s) = C(x)$ except with negligible probability.

5.2. Proof of Security

The proof of security is simlar to that of the receiver deniable FE scheme of the conference version [21]. The main point here is to show that public deniability interact nicely with receiver deniable and allows to obtain bi-deniability.

More specifically, we prove the following main theorems.

Theorem 5.2. If PDFE is IND-Secure, then DenFE is IND-Secure as well.

The proof of Theorem 5.2 is straightforward and we omit it.

Theorem 5.3. If PDFE is $(\mathsf{poly}, 1, \mathsf{poly})$-IND-Secure then DenFE is a (n_c, poly)-*bideniable* in the sense of Definition 2.6, for any constant n_c.

Proof. We prove security via a sequence of hybrid experiments. To do so, we will make use of the same simulation receiver faking algorithm of that used in the security for the receiver deniable FE of the conference version. We report it here for completeness.

$\mathsf{Sim.RecFake}^{\mathsf{PDFE.KeyGen}(\mathsf{PDFE.Msk},\cdot)}(C, \mathbf{x}, \mathbf{s})$

The algorithm takes in input a circuit C, messages $\mathbf{x} = (x_1, \ldots, x_\ell)$, strings $\mathbf{s} = (s_1, \ldots, s_\ell)$ each in $\{0,1\}^\lambda$, and oracle access to the PDFE key generation algorithm. Then, for each $i \in [\ell]$, the algorithm chooses random t_i and t'_i in $\{0,1\}^{l(\lambda)}$ and distinguishes between the following two case:

- If $C(x_i) = 1$, it sets $z_i = f_{s_i}(t_i)$ and chooses random $z'_i \in \{0,1\}^{L(\lambda)}$.
- If $C(x_i) = 0$, it sets $z'_i = f_{s_i}(t'_i)$ and chooses random $z_i \in \{0,1\}^{L(\lambda)}$.

Finally, the algorithm computes $\mathsf{PDFE.Sk}_C = \mathsf{PDFE.KeyGen}(\mathsf{PDFE.Msk}, \mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'})$, and returns secret key $\mathsf{Sk}_C = (\mathbf{t}, \mathbf{z}, \mathbf{t}', \mathbf{z}', \mathsf{PDFE.Sk}_C)$.

We are now ready to describe the hybrids. The change between the presented hybrid and the previous will be denoted by boxing the modified parts.

1. Hybrid H_1: This is the real experiment RealDenExp where the sender-coerce oracle, let us call it $\mathcal{E}_1^\star$, and the receiver-coerce oracle, let us call it $\mathcal{K}_1^\star$, are the following:

$\mathcal{E}_1^\star(\mathbf{x}, \mathbf{y})$	$\mathcal{K}_1^\star(C, \mathbf{Ct}, \mathbf{x})$
$(s_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$	$\mathsf{Sk}_C \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, C)$;
$(\mathsf{Ct}_i \leftarrow \mathsf{PDFE.Enc}(\mathsf{Mpk}, (x_i, s_i); r_i))_{i\in[n_c]}$	**Output:** Sk_k
Output: $((\mathsf{Ct}_i), (s_i, r_i))$	

Notice that $\mathcal{E}_1^\star$ is exactly $\mathcal{E}_1$ with the only difference that we have unrolled the call to the RecDenFE encryption algorithm for the sake of clarity, and $\mathcal{K}_1^\star = \mathcal{K}_1$.

2. Hybrid H_2: This is the same as H_1 except that the sender-coerce oracle and receiver-coerce oracle are modified as follows:

$\mathcal{E}_2^\star(\mathbf{x}, \mathbf{y})$
$(s_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$
$\boxed{(s'_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}}$
$(\mathsf{Ct}_i \leftarrow \mathsf{PDFE.Enc}(\mathsf{Mpk}, (x_i, s_i); r_i))_{i\in[n_c]}$
Output: $((\mathsf{Ct}_i), (s_i, r_i))$

$\mathcal{K}_2^\star(C, \mathbf{Ct}, \mathbf{x})$
$\boxed{\mathsf{Sk}_C \leftarrow \mathsf{Sim.RecFake}^{\mathsf{PDFE.KeyGen}(\mathsf{PDFE.Msk},\cdot)}(C, \mathbf{x}, \mathbf{s}')}$
Output: Sk_C

where $\mathbf{s}' = (s'_1, \ldots, s'_{n_c})$ is the randomness sampled by $\mathcal{E}_2^\star$.

3. Hybrid H_3: This is the same as H_2 except that the sender-coerce oracle is modified as follows.

$\mathcal{E}_3^\star(\mathbf{x}, \mathbf{y})$
$(s_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$
$(s'_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$
$(\mathsf{Ct}_i \leftarrow \mathsf{PDFE.Enc}(\mathsf{Mpk}, (x_i, s_i)))_{i\in[n_c]}$
$\boxed{(e_i \leftarrow \mathsf{PDFE.Explain}(\mathsf{Ct}_i, (x_i, s_i)))_{i\in[n_c]}}$
Output: $((\mathsf{Ct}_i), (s_i, e_i)$

$\mathcal{K}_3^\star(C, \mathbf{Ct}, \mathbf{x})$
$\mathsf{Sk}_C \leftarrow \mathsf{Sim.RecFake}^{\mathsf{PDFE.KeyGen}(\mathsf{PDFE.Msk},\cdot)}(C, \mathbf{x}, \mathbf{s}')$
Output: Sk_C

4. Hybrid H_4: This is the same as H_3 except that the sender-coerce oracle is modified as follows.

$\mathcal{E}_4^\star(\mathbf{x}, \mathbf{y})$
$(s_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$
$(s'_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$
$\boxed{(\mathsf{Ct}_i \leftarrow \mathsf{PDFE.Enc}(\mathsf{Mpk}, (y_i, s'_i)))_{i\in[n_c]}}$
$(e_i \leftarrow \mathsf{PDFE.Explain}(\mathsf{Ct}_i, (x_i, s_i)))_{i\in[n_c]}$
Output: $((\mathsf{Ct}_i), \emptyset)$

$\mathcal{K}_4^\star(C, \mathbf{Ct}, \mathbf{x})$
$\mathsf{Sk}_C \leftarrow \mathsf{Sim.RecFake}^{\mathsf{PDFE.KeyGen}(\mathsf{PDFE.Msk},\cdot)}(C, \mathbf{x}, \mathbf{s})$
Output: Sk_C

Finally, notice that H_4 is exactly the faking experiment $\mathsf{FakeDenExp}$ where $\mathcal{E}_2 = \mathcal{E}_4^\star$ and $\mathcal{K}_2 = \mathcal{K}_4^\star$.

We now show that the relevant distinguishing probabilities between adjacent hybrids are negligible, which completes the proof.

Indistinguishability of H_1 and H_2: This step is exactly the same of the proof for the receiver deniable FE of the conference version.

Indistinguishability of H_2 and H_3: To prove indistinguishability we use the following sequence of hybrid experiments.

- Hybrid $H_{2,j}$, for $1 \le j \le n_c + 1$: This is the same as H_2 except that the following new sender-coercer oracle is used:

$\mathcal{E}_{2,j}^\star(\mathbf{x}, \mathbf{y})$
$(s_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$
$(s'_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$

$\mathsf{Ct}_i \leftarrow \mathsf{PDFE.Enc}(\mathsf{Mpk}, (x_i, s_i); r_i)$
Then, for $i < j$, $e_i \leftarrow \mathsf{PDFE.Explain}(\mathsf{Ct}_i, (x_i, s_i)))_{i\in[n_c]}$,
Output: $((\mathsf{Ct}_i), ((s_1, e_1), \ldots, (s_{j-1}, e_{j-1}), (s_j, r_j), \ldots, (s_{n_c}, r_{n_c})))$

Then, notice that $H_2 = H_{2,1}$ and $H_3 = H_{2,n_c+1}$. Thus, it is sufficient to prove that $H_{2,k}$

is computational indistinguishable from $H_{2,k+1}$. This can be reduced to the indistinguishability of explanation of PDFE. In fact, notice that $H_{1,k+1}$ is the same as $H_{1,k}$ except that the explain algorithm is used to fake sender randomness r_k. Notice that, in this step, we use the fact that the indistinguishability of explanation of PDFE scheme holds also when the adversary is given the master secret key of the underlying functional encryption scheme.

Thus, for sake of contradiction, suppose there exists a distinguisher $\mathcal{D}$ and adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ for which $H_{2,k}$ and $H_{2,k+1}$ are not computationally indistinguishable. Then $\mathcal{A}$ and $\mathcal{D}$ can be used to construct a successful IND-EXPL adversary $\mathcal{B}$ for PDFE. Specifically, $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ does the following.

- $\mathcal{B}_0$ on input PDFE master public key Mpk, and master secret key of the PDFE's underlying functional encryption FE.Msk, runs $\mathcal{A}_0$ on input master public key Mpk and answers $\mathcal{A}_0$'s queries to $\mathcal{O}_1$ and $\mathcal{O}_2$ by using Mpk and FE.Msk.

  Eventually, $\mathcal{A}_0$ outputs $\mathbf{x}^\star = (x_1^\star, \ldots, x_{n_c}^\star)$, $\mathbf{y}^\star = (y_1^\star, \ldots, y_{n_c}^\star)$ and its state $\mathtt{st}$.

  Then, $\mathcal{B}_0$ chooses random $s \in \{0,1\}^n$ and outputs $(x_k^\star, s^\star)$, and put in its state the state of $\mathcal{A}_0$ and its entire computation.
- $\mathcal{B}_1$ on input ciphertext $\mathsf{Ct}^\star$, encryption of $(x_k^\star, s^\star)$, randomness $r^\star$, which is the real randomness used to generate $\mathsf{Ct}^\star$ or the randomness obtained by the explain algorithm, and state $\mathtt{st}$, does the following: $\mathcal{B}_1$, chooses $(s_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$ and $(s_i' \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$, then, for $i \in [n_c] \setminus \{k\}$, sets $\mathsf{Ct}_i^\star = \mathsf{Encrypt}(\mathsf{Mpk}, (x_i^\star, s_i))$, and for index k, $\mathcal{B}_1$ sets $\mathsf{Ct}_k^\star = \mathsf{Ct}^\star$. Finally, for $i < j$, $\mathcal{B}_1$ sets $e_i \leftarrow \mathsf{PDFE.Explain}(\mathsf{Ct}_i, (x_i, s_i)))_{i\in[n_c]}$,

  Finally, $\mathcal{B}_1$ runs $\mathcal{A}_1$ on input challenge ciphertexts and sender randomness $\mathsf{Ct}_1^\star, \ldots, \mathsf{Ct}_{n_c}^\star$ and sender randomness $((s_1, e_1), \ldots, (s_{j-1}, e_{j-1}), (s_j, r_j), \ldots, (s_{n_c}, r_{n_c}))$, and answers $\mathcal{A}_1$'s queries to $\mathcal{O}_1$ and $\mathcal{O}_2$ and to the receiver-coerce oracle $\mathcal{K}$, by using Mpk and FE.Msk.

  Eventually, $\mathcal{A}_1$ returns its output and $\mathcal{B}_1$ passes it to the distinguisher $\mathcal{D}$ and returns $\mathcal{D}$'s output as its own output.

Now notice that if $r^\star$ is the real randomness used to encrypt $(x_k^\star, s^\star)$ then $\mathcal{B}$ is simulating $H_{2,k}$. On the other hand if $r^\star$ has been generated by the explain algorithm then $\mathcal{B}$ is simulating $H_{2,k+1}$.

Indistinguishability of H_3 and H_4: This step is exactly the same of the proof for the receiver deniable FE of the conference version, here we proof that H_2 is computational indistinguishable from H_3 under the IND security of the underlying functional encryption scheme.

This concludes the proof. □

6. Open problems and future work

Our work leaves open the problem of a construction of a multidistributional deniable FE for general functionalities that avoid the use of diO. It is also worthy to investigate whether our techniques can be used to add deniability to other flavors of FE, e.g., [43, 11, 14, 33].

7. Acknowledgments

Vincenzo Iovino is supported by a CORE Junior grant (no. 11299247) of the Luxembourg National Research Fund.

8. References

[1] S. Agrawal, S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption: New perspectives and lower bounds. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 500–518. Springer, Aug. 2013.

[2] S. Agrawal, V. Koppula, and B. Waters. Impossibility of simulation secure functional encryption even with random oracles. *IACR Cryptology ePrint Archive*, 2016:959, 2016.

[3] P. Ananth, D. Boneh, S. Garg, A. Sahai, and M. Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. `http://eprint.iacr.org/2013/689`.

[4] D. Apon, X. Fan, and F. Liu. Bi-deniable inner product encryption from LWE. *IACR Cryptology ePrint Archive*, 2015:993, 2015.

[5] D. Apon, X. Fan, and F. Liu. Deniable attribute based encryption for branching programs from LWE. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 299–329, 2016.

[6] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, Aug. 2001.

[7] M. Barbosa and P. Farshim. On the semantic security of functional encryption schemes. In *Public Key Cryptography*, pages 143–161, 2013.

[8] M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In A. Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 1–35. Springer, Apr. 2009.

[9] M. Bellare and A. O'Neill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In M. Abdalla, C. Nita-Rotaru, and R. Dahab, editors, *CANS 13: 12th International Conference on Cryptology and Network Security*, volume 8257 of *Lecture Notes in Computer Science*, pages 218–234. Springer, Nov. 2013.

[10] R. Bendlin, J. B. Nielsen, P. S. Nordholt, and C. Orlandi. Lower and upper bounds for deniable public-key encryption. In D. H. Lee and X. Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 125–142. Springer, Dec. 2011.

[11] C. Blundo, V. Iovino, and G. Persiano. Predicate encryption with partial public keys. In S.-H. Heng, R. N. Wright, and B.-M. Goi, editors, *CANS 10: 9th International Conference on Cryptology and Network Security*, volume 6467 of *Lecture Notes in Computer Science*, pages 298–313. Springer, Dec. 2010.

[12] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, May 2004.

[13] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, Aug. 2001.

[14] D. Boneh, A. Raghunathan, and G. Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 461–478. Springer, Aug. 2013.

[15] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In Y. Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, Mar. 2011.

[16] D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In K. Sako and P. Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 280–300. Springer, Dec. 2013.

[17] E. Boyle, K.-M. Chung, and R. Pass. On extractability obfuscation. In Y. Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73. Springer, Feb. 2014.

[18] E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In H. Krawczyk, editor, *PKC 2014: 17th International Workshop on Theory and Practice in Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 501–519. Springer, Mar. 2014.

[19] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In B. S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 90–104. Springer, Aug. 1997.

[20] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on Theory of Computing*, pages 639–648. ACM Press, May 1996.

[21] A. D. Caro, V. Iovino, and A. O'Neill. Deniable functional encryption. In *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I*, pages 196–222, 2016.

[22] I. Damgård and J. B. Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In M. Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 432–450. Springer, Aug. 2000.

[23] A. De Caro and V. Iovino. On the power of rewinding simulators in functional encryption. *Designs, Codes and Cryptography*, pages 1–27, 2016.

[24] A. De Caro, V. Iovino, A. Jain, A. O'Neill, O. Paneth, and G. Persiano. On the achievability of simulation-based security for functional encryption. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 519–535. Springer, Aug. 2013.

[25] C. Dwork, M. Naor, O. Reingold, and L. J. Stockmeyer. Magic functions. In *40th Annual Symposium on Foundations of Computer Science*, pages 523–534. IEEE Computer Society Press, Oct. 1999.

[26] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49. IEEE Computer Society Press, Oct. 2013.

[27] S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters. Attribute-based encryption for circuits from multilinear maps. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 479–499. Springer, Aug. 2013.

[28] S. Garg, C. Gentry, S. Halevi, and M. Zhandry. Functional encryption without obfuscation. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, pages 480–511, 2016.

[29] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science*, pages 464–479. IEEE Computer Society Press, Oct. 1984.

[30] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[31] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 162–179. Springer, 2012.

[32] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 545–554. ACM Press, June 2013.

[33] V. Iovino, Q. Tang, and K. Żebrowski. On the power of public-key function-private functional encryption. In *Cryptology and Network Security - 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings*, pages 585–593, 2016.

[34] V. Iovino and K. Żebrowski. Simulation-based secure functional encryption in the random oracle model. In *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*, pages 21–39, 2015.

[35] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In N. P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer, Apr. 2008.

[36] A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 13: 20th Conference on Computer and Communications Security*, pages 669–684. ACM Press, Nov. 2013.

[37] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437. ACM Press, May 1990.

[38] T. Okamoto and K. Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 591–608. Springer, Apr. 2012.

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

[40] A. O'Neill, C. Peikert, and B. Waters. Bi-deniable public-key encryption. In P. Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 525–542. Springer, Aug. 2011.

[41] A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In D. B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484. ACM Press, May / June 2014.

[42] A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, Aug. 1984.

[43] E. Shen, E. Shi, and B. Waters. Predicate privacy in encryption systems. In O. Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 457–473. Springer, Mar. 2009.

[44] B. Waters. A punctured programming approach to adaptively secure functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 678–697, 2015.