**Receiver and Sender Deniable Functional Encryption**

Angelo De Caro¹, Vincenzo Iovino²*, Adam O’Neill³

¹IBM Research, Zurich
²University of Luxembourg
³Georgetown 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 \( C_t \) 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 \( C_t \)
that encrypts some string \( x \). Informally, receiver-deniable FE allows the sender to produce “fake”
secret key \( Sk_k \) that makes equivocate \( C_t \) 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
coerencer 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 \textit{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 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 \cite{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. \cite{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 \( \text{Trap}[C] \) 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, \( \text{Trap}[C] \), 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 \( \text{Trap}[C] \) returns 0, if \( s = r_2 \) then it returns 1, otherwise \( \text{Trap}[C] \) 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 \( \text{Trap}[C] \), 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 \cite{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 \cite{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 \cite{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 \( n_k = poly \).
"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 \( \text{Trap}[C] \), 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 \( \text{Ct}' \), has to be linked to the corresponding \( \text{Trap}[C] \) 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. \( \text{Trap}[C] \) then will take in input also \( \text{Ct}' \) 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 \( \text{Ct}' \) 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 \( \text{Trap}[C] \) that takes in input the double encryption of \( x \) and the double encryption of the trapdoor values related to \( \text{Trap}[C] \). Intuitively, differing-input obfuscation is required because there are certain \( \text{Ct}' \) that allows to discriminate, for example, which secret key \( \text{Trap}[C] \) 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 \((\text{poly}, \text{poly})\)-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).
this way we achieve \((n_c, \text{poly})\) 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 \((q_1, \ell, q_2)\)-SIM security denotes SIM-security where the adversary is allowed to make at most \(q_1\) non-adaptive key queries, \(\ell\) 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 \((n_c, \text{poly})\)-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 \((q_1, \ell, \text{poly})\)-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 \(F_n : K_n \times X_n \rightarrow \Sigma\) where \(K_n\) is the key space for parameter \(n\), \(X_n\) is the message space for parameter \(n\) and \(\Sigma\) is the output space. We will refer to functionality \(F\) as a function from \(F : K \times X \rightarrow \Sigma\) 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
Definition 2.2. [Functional Encryption Scheme] A functional encryption scheme for functionality $F$ defined over $(K, X)$ is a tuple $FE = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ of 4 algorithms with the following syntax:

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

In addition we make the following correctness requirement: for all $(Mpk, Msk) \leftarrow \text{Setup}(\lambda^1, 1^n)$, 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: $\text{Enc}(Mpk, x) = F(k, x)$ whenever $F(k, x) \neq \perp$ where $Sk \leftarrow \text{KeyGen}(Msk, k)$ and $Ct \leftarrow \text{Enc}(Mpk, x)$.

Definition 2.3. [Circuit Functionality] The 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 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 $FE = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ for functionality $F$ defined over $(K, X)$ is formalized by means of the following game $\text{IND}_{\mathcal{A}}^{\text{FE}}$ between an adversary $\mathcal{A} = (A_0, A_1)$ and a challenger $C$.

\[
\text{IND}_{\mathcal{A}}^{\text{FE}}(\lambda^1) = \begin{cases} 
1. \ C \text{ generates } (Mpk, Msk) \leftarrow \text{Setup}(\lambda^1) \text{ and runs } A_0 \text{ on input } Mpk; \\
2. \ A_0, \text{ during its computation, issues } q_1 \text{ non-adaptive key-generation queries. } C \text{ on input key } k \in K \text{ computes } Sk = \text{KeyGen}(Msk, k) \text{ and sends it to } A_0. \\
\quad \text{When } A_0 \text{ stops, it outputs two challenge messages vectors, of length } \ell, x_0, x_1 \in X^{\ell} \text{ and its internal state } \sigma. \\
3. \ C \text{ picks } b \in \{0, 1\} \text{ at random, and, for } i \in \ell, \text{ computes the challenge ciphertexts } Ct_i = \text{Enc}(Mpk, x_i) \text{. Then } C \text{ sends } (Ct_i)_{i \in \ell} \text{ to } A_1 \text{ that resumes its computation from state } \sigma. \\
4. \ A_1, \text{ during its computation, issues } q_2 \text{ adaptive key-generation queries. } C \text{ on input key } k \in K \text{ computes } Sk = \text{KeyGen}(Msk, k) \text{ and sends it to } A_1. \\
5. \ A_1 \text{ stops, it outputs } b'. \\
6. \textbf{Output: if } b = b', \text{ for each } i \in \ell, |x'_i| = |x_i|, \text{ and } F(k, x_0) = F(k, x_1) \text{ for each } k \text{ for which } A \text{ has issued a key-generation query, then output 1 else output 0.} \end{cases}
\]

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

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

Definition 2.4. We say that $FE$ is $(q_1, \ell, q_2)$-indistinguishably secure ($(q_1, \ell, q_2)$-IND-Secure, for short) where $q_1 = q_1(\lambda), \ell = \ell(\lambda)q_2 = q_2(\lambda)$ are polynomials in the security parameter $\lambda$ 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 $\ell$, 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.

\footnote{See \cite{ref} 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 \(\lambda\) that are fixed a priori, is made up of the algorithms \(\text{RecDenFE} = (\text{Setup}, \text{Enc}, \text{KeyGen}, \text{Dec})\) of a standard FE scheme for \(F\) (Definition 2.2) and in addition the following algorithm:

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

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

**Definition 2.6.** [Receiver-Deniable] 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.

<table>
<thead>
<tr>
<th>RealRecDenExp(_\mathcal{A})^{\text{RecDenFE}}(1^\lambda)</th>
<th>FakeRecDenExp(_\mathcal{A})^{\text{RecDenFE}}(1^\lambda)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\text{Mpk}, \text{Msk}) \leftarrow \text{Setup}(1^\lambda); (x^<em>, y^</em>, st) \leftarrow \mathcal{A}_0^{O_1, O_2}(\text{Mpk}); (\text{Ct}<em>i^* \leftarrow \text{Enc}(\text{Mpk}, x_i; r_i))</em>{i \in [n_c]}; \text{Output: } \mathcal{A}_1^{O_1, O_2, \mathcal{K}_1}(\text{Ct}^<em>, x^</em>) (\text{Ct}^*, st))</td>
<td>((\text{Mpk}, \text{Msk}) \leftarrow \text{Setup}(1^\lambda); (x^<em>, y^</em>, st) \leftarrow \mathcal{A}_0^{O_1, O_2}(\text{Mpk}); (\text{Ct}<em>i^* \leftarrow \text{Enc}(\text{Mpk}, y_i; r_i))</em>{i \in [n_c]}; \text{Output: } \mathcal{A}_1^{O_1, O_2, \mathcal{K}_2}(\text{Ct}^<em>, x^</em>) (\text{Ct}^*, st))</td>
</tr>
</tbody>
</table>

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

All the oracles declared above are defined as follows:

<table>
<thead>
<tr>
<th>(\mathcal{K}_1(k, \text{Ct}, x))</th>
<th>(\mathcal{K}_2(k, \text{Ct}, x))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{Sk}_k \leftarrow \text{KeyGen}(\text{Msk}, k)); Output: (\text{Sk}_k)</td>
<td>(\text{Sk}_k \leftarrow \text{RecFake}(\text{Msk}, k, \text{Ct}, x)); Output: (\text{Sk}_k)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(\mathcal{O}_1(k, x, y))</th>
<th>(\mathcal{O}_2(k, x, y))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{Ct} \leftarrow \text{Enc}(\text{Mpk}, x; r)); (\text{Sk}_k \leftarrow \text{KeyGen}(\text{Msk}, k)); Output: ((\text{Ct}, \text{Sk}_k))</td>
<td>(\text{Ct} \leftarrow \text{Enc}(\text{Mpk}, y; r)); (\text{Sk}_k \leftarrow \text{RecFake}(\text{Msk}, k, \text{Ct}, x)); Output: ((\text{Ct}, \text{Sk}_k))</td>
</tr>
</tbody>
</table>

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 \((k, \text{Ct}^*, x)\) for some \(x\) issued to \(\mathcal{K}_1\) and there is no query \((k, x, y)\) issued to \(\mathcal{O}_2\) and at same time a query \((k, \text{Ct}^*, x)\) for some \(x\) 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 \(k^*\), neither \(\mathcal{A}_0\) nor \(\mathcal{A}_1\) queries \(k^*\) to either of their oracles \(\mathcal{O}_1, \mathcal{O}_2\); i.e., they do not make any query \((k^*, x, y)\) for some \(x, y\) to \(\mathcal{O}_1\) or \(\mathcal{O}_2\).

3. For each key \(k\) different from any of the challenge keys \(k^*_i\) queried by \(\mathcal{A}_0\) and \(\mathcal{A}_1\) to oracles \(\mathcal{O}_1\) or \(\mathcal{O}_2\), it holds that \(F(k; x^*) = F(k, y^*)\).
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 \(\lambda\) that are fixed a priori, is made up of the algorithms \(\text{MDRecDenFE} = (\text{Setup}, \text{Enc}, \text{KeyGen}, \text{Dec})\) of a standard FE scheme for \(F\) (Definition 2.2) and in addition the following two algorithms:

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

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 \(\mathcal{K}_1, \mathcal{K}_2\), the following two experiments are computationally indistinguishable.

<table>
<thead>
<tr>
<th>RealMDRecDenExp_{\mathcal{A}0}^{\text{MDRecDenFE}(1^\lambda)}</th>
<th>FakeMDRecDenExp_{\mathcal{A}1}^{\text{MDRecDenFE}(1^\lambda)}</th>
</tr>
</thead>
<tbody>
<tr>
<td>((x^<em>, y^</em>, \text{st}) \leftarrow \mathcal{A}_0(1^\lambda); )</td>
<td>((x^<em>, y^</em>, \text{st}) \leftarrow \mathcal{A}_0(1^\lambda); )</td>
</tr>
<tr>
<td>((\text{Mpk}, \text{Msk}) \leftarrow \text{Setup}(1^\lambda); )</td>
<td>((\text{Mpk}, \text{Msk}) \leftarrow \text{Setup}(1^\lambda); )</td>
</tr>
<tr>
<td>((\text{Ct}<em>i^* \leftarrow \text{Enc}(\text{Mpk}, x_i; r_i))</em>{i \in [n_c]}; )</td>
<td>((\text{Ct}<em>i^* \leftarrow \text{Enc}(\text{Mpk}, y_i; r_i))</em>{i \in [n_c]}; )</td>
</tr>
<tr>
<td>\text{Output: } \mathcal{A}_1^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{K}_1, \text{Ct}^<em>, x^</em>}(\text{Mpk}, \text{Ct}^*, \text{st})</td>
<td>\text{Output: } \mathcal{A}_1^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{K}_2, \text{Ct}^<em>, x^</em>}(\text{Mpk}, \text{Ct}^*, \text{st})</td>
</tr>
</tbody>
</table>

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

All the oracles declared above are defined as follows:

\[
\begin{align*}
\mathcal{K}_1(k, \text{Ct}, x) & \quad \mathcal{K}_2(k, \text{Ct}, x) \\
\text{Sk}_k & \leftarrow \text{KeyGen}(\text{Msk}, k); \\
\text{Output: } & \text{Sk}_k \\
\text{Sk}_k' & \leftarrow \text{RecFake}(\text{Sk}_k, \text{Fk}_k, \text{Ct}, x); \\
\text{Output: } & \text{Sk}_k' \\
\mathcal{O}_1(k, x, y) & \quad \mathcal{O}_2(k, x, y) \\
\text{Ct} & \leftarrow \text{Enc}(\text{Mpk}, x; r); \\
\text{Sk}_k & \leftarrow \text{KeyGen}(\text{Msk}, k); \\
\text{Output: } & (\text{Ct}, \text{Sk}_k) \\
\text{Sk}_k' & \leftarrow \text{RecFake}(\text{Sk}_k, \text{Fk}_k, \text{Ct}, x); \\
\text{Output: } & (\text{Ct}, \text{Sk}_k')
\end{align*}
\]

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 \((k, \text{Ct}^*, x)\) for some \(x\) issued to \(\mathcal{K}_1\) and there is no query \((k, x, y)\) issued to \(\mathcal{O}_2\) and at same time a query \((k, \text{Ct}^*, x)\) for some \(x\) 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 \(k^*\), neither \(\mathcal{A}_0\) nor \(\mathcal{A}_1\) queries \(k^*\) to either of their oracles \(\mathcal{O}_1, \mathcal{O}_2\); i.e., they do not make any query \((k^*, x, y)\) for some \(x, y\) to \(\mathcal{O}_1\) or \(\mathcal{O}_2\).
3. For each key $k$ different from any of the challenge keys $k_i^*$ queried by $A$ to oracles $O_1$ or $O_2$, it holds that $F(k, x^*) = F(k, y^*)$.

**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 $PDE = (\text{Setup}, \text{Enc}, \text{Dec}, \text{Explain})$ with the following semantics:

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

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

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

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 $PDE = (\text{Setup}, \text{Enc}, \text{Dec}, \text{Explain})$ is formalized by means of the following game $\text{IND-EXPL}_{A}^{\text{PDE}}$ between an adversary $A = (A_0, A_1)$ and a challenger $C$.

<table>
<thead>
<tr>
<th>$\text{IND-EXPL}_{A}^{\text{PDE}}(1^\lambda)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $C$ generates $(Mpk, Sk) \leftarrow \text{Setup}(1^\lambda)$ and runs $A_0$ on input $Mpk$;</td>
</tr>
</tbody>
</table>
| 2. When $A_0$ stops, it outputs a *single* message $x \in X$ and its internal state $st$. $C$ creates $Ct \leftarrow \text{Enc}(Mpk, x; u)$ for random $u$ and it computes $e \leftarrow \text{Explain}(Mpk, Ct, x; r)$ for random $r$. Then, $C$ flips a coin $b \leftarrow \{0, 1\}$. If $b = 0$, $C$ sends $(Ct, u)$ to $A_1$ that resumes its computation from state $st$. Otherwise, if $b = 1$, $C$ sends $(Ct, e)$.
| 3. When $A_1$ stops, it outputs $b'$.
| 4. Output: if $b = b'$, then output 1 else output 0. |

The advantage of adversary $A$ in the above game is defined as

$$\text{Adv}_{A}^{\text{PDE,IND-EXPL}}(1^\lambda) = \Pr[\text{IND-EXPL}_{A}^{\text{PDE}}(1^\lambda) = 1] - 1/2$$
Definition 2.11. We say that PDE has indistinguishably of explanation if all probabilistic polynomial time adversaries $A$ 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 $E$ 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 $E$.

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 $OR_{I_1, I_2}(x_1, x_2) = 1$ 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 $AND_{I_1, I_2}(x_1, x_2) = 1$ 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 encryption 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 \textit{trapdoor boolean formula} \(\text{Trap}[f]_{r_0, r_1}\) on \((n + \ell)\)-bit inputs as follows: \(\text{Formula}_\text{Trap}[f]_{r_0, r_1}(x, s) := (s = r_1) \lor [f(x) \land \neg(s = r_0)]\), 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 \(\text{FE} = (\text{FE.Setup}, \text{FE.Enc}, \text{FE.KeyGen}, \text{FE.Eval})\) be the functional encryption scheme for the functionality \(\text{Boolean Formulae}\). For any constant \(\ell > 3\), we define our \textit{receiver deniable functional encryption scheme} \(\text{RecDenFE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{RecFake})\) for Boolean formulae as follows.

- Setup\((1^\lambda, 1^n, 1^m)\), for each \(i \in [m]\), runs \(\text{FE.Setup}(1^\lambda, 1^{n+\ell})\) to get the pair \((\text{FE.Mpk}_i, \text{FE.Msk}_i)\).

- Then, the master public key is \(\text{Mpk} = (\text{FE.Mpk})_{i \in [m]}\) and the master secret key is \(\text{Msk} = (\text{FE.Msk})_{i \in [m]}\).

- The algorithm returns the pair (Mpk, Msk).

- \(\text{Enc}(\text{Mpk}, x)\), on input master public key \(\text{Mpk} = (\text{FE.Mpk})_{i \in [m]}\) and message \(x \in \{0, 1\}^n\), for each \(i \in [m]\), chooses a random \(s_i \in \{0, 1\}^\ell\) and sets \(\text{Ct}_i = \text{FE.Enc}(\text{FE.Mpk}_i, (x, s_i))\). The algorithm returns the ciphertext \(\text{Ct} = (\text{Ct}_i)_{i \in [m]}\).

- \(\text{KeyGen}(\text{Msk}, f)\), on input master secret key \(\text{Msk} = (\text{FE.Msk})_{i \in [m]}\) and a \(n\)-input Boolean formula \(f\), for each \(i \in [m]\), chooses two random strings \(r_{0i}, r_{1i} \in \{0, 1\}^\ell\), such that \(r_{0i} \neq r_{1i}\), and computes secret key \(\text{FE.Sk}_f^i = \text{FE.KeyGen}(\text{FE.Msk}_i, \text{Trap}[f]_{r_{0i}, r_{1i}})\). The algorithm returns the secret key \(\text{Sk}_f = (r_{0i}, r_{1i}, \text{FE.Sk}_f^i)_{i \in [m]}\).

- Dec\((\text{Mpk}, \text{Ct}, \text{Sk}_f)\), on input master public key \(\text{Mpk} = (\text{FE.Mpk})_{i \in [m]}\), \(\text{Ct} = (\text{Ct}_i)_{i \in [m]}\) and secret key \(\text{Sk}_f = (r_{0i}, r_{1i}, \text{FE.Sk}_f^i)_{i \in [m]}\) for Boolean formula \(f\), for \(i \in [m]\), computes Boolean value \(b_i = \text{FE.Eval}(\text{FE.Mpk}_i, \text{Ct}_i, \text{FE.Sk}_f^i)\), and returns as output the Boolean value on which the majority of \(b_i\)’s have agreed on.

- \(\text{RecFake}(\text{Msk}, f, \text{Ct}, x')\), on input the master secret key \(\text{Msk} = (\text{FE.Msk})_{i \in [m]}\), an \(n\)-input Boolean formula \(f\), ciphertext \(\text{Ct} = (\text{Ct}_i)_{i \in [m]}\) and message \(x'\), for all \(i \in [m]\), the algorithm extracts \(s_i\) from \(\text{Ct}_i\) by using \(\text{FE.Msk}_i\). Now, \(\text{RecFake}\) chooses the \(r_{0i}\)’s and \(r_{1i}\)’s by following a binomial distribution with number of trials equals to \(m\) and success probability \(p = (1 - 2^{-\ell})\). Specifically, \(\text{RecFake}\) 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 \(\text{RecFake}\) sets \(r_{0i}' = s_i\) and \(r_{1i}' = b_i\) to a random value different from \(s_i\). Otherwise, \(\text{RecFake}\) sets \(r_{1i}' - b_i\) to a random value different from \(s_i\). Finally, \(\text{RecFake}\) computes secret key \(\text{FE.Sk}_f^i = \text{FE.KeyGen}(\text{FE.Msk}_i, \text{Trap}[f]_{r_{0i}', r_{1i}'})\), and returns the secret key \(\text{Sk}_f = (r_{0i}', r_{1i}', \text{FE.Sk}_f^i)_{i \in [m]}\) as faking key.

Correctness. Notice that for any \(i \in [m]\), the probability that \(s_i = r_{0i}' \lor s_i = r_{1i}'\) is at most \(2^{-\ell+1}\).
Thus the output of the decryption is correct, i.e. \( \text{Trap}[f]_{\mathcal{T}_0 \oplus \mathcal{T}_1}(x, s_i) = f(x) \), 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 \( \text{RecDenFE} \) hold with overwhelming probability.

**Security.** The proof that \( \text{RecDenFE} \) 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 \( n_k = \text{poly} \) in this case, however, because we cannot use symmetric encryption (at least in a straightforward way).

## 4. Multi-Distributional Receiver Deniable FE from diO

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 \( \text{Trap}[C] \) 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 \( \mathcal{E} = (\mathcal{E}.\text{Setup}, \mathcal{E}.\text{Enc}, \mathcal{E}.\text{Dec}) \) with perfect correctness, a differing-inputs obfuscator \( \text{diO} \), an SSS-NIZK proof system NIZK = (NIZK.Setup, NIZK.Prove, NIZK.Verify, NIZK.Sim) and a one-way function \( f \), we define our multi-distributional receiver deniable functional encryption \( \text{MDRecDenFE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{DenKeyGen}, \text{RecFake}, \text{Dec}) \) as follows:

1. **Setup(\( \lambda \))** takes as input the **security parameter** \( \lambda \) and computes the following: For \( i \in [4] \), \((\text{pk}_i, \text{sk}_i) \leftarrow \mathcal{E}.\text{Setup}(\lambda) \). Then, \( \text{crs} \leftarrow \text{NIZK.Setup}(\lambda) \). The algorithm sets \( \text{Mpk} = ((\text{pk}_i)_{i \in [4]}, f, \text{crs}) \), \( \text{Msk} = ((\text{sk}_i)_{i \in [4]}) \).

2. **KeyGen(Msk, \( C \))** takes as input **master secret key** \( \text{Msk} = ((\text{sk}_i)_{i \in [4]}) \), and **circuit** \( C \), and does the following: Computes common reference string \( \text{crs}' \leftarrow \text{NIZK.Setup}(\lambda) \). Then, sample random \( z \) in the domain of \( f \) and set \( t = f(z) \) and compute \( \text{Ct}' := (\text{ct}_3, \text{ct}_4, \pi_2) \), where \( \text{ct}_3 \leftarrow \mathcal{E}.\text{Enc}(\text{pk}_3, (z, 0^n, 0^\lambda); r_3) \) and \( \text{ct}_4 \leftarrow \mathcal{E}.\text{Enc}(\text{pk}_4, (z, 0^n, 0^\lambda); r_4) \), and \( \pi_2 \) is a NIZK proof of Equation 2. Finally, the algorithm computes a differing-input obfuscation \( \text{diO}_{\text{Trap}_{1,3}} \) for the trapdoor circuit \( \text{Trap}_{1,3}[C, \text{crs}, \text{crs}', \text{sk}_1, \text{sk}_3, f, t] \). The algorithm outputs **secret key** for the circuit \( C \), \( \text{Sk}_C = (\text{diO}_{\text{Trap}_{1,3}}, t, \text{Ct}') \).

3. **DenKeyGen(Msk, \( C \))** takes as input \( \text{Msk} = ((\text{sk}_i)_{i \in [4]}) \) and **circuit** \( C \), and computes \( \text{Sk}_C = \text{KeyGen(Msk, C)} \). The algorithm outputs \( \text{Sk}_C \) as the **secret key** for the circuit \( C \) and \( \text{Fk}_C = z \).

4. **Enc(Mpk, \( x \))**, on input **master public key** \( \text{Mpk} = ((\text{pk}_i)_{i \in [4]}, f, \text{crs}) \) and messages \( x \in X_n \), computes \( \text{Ct} = (\text{ct}_1, \text{ct}_2, \pi_1) \), where \( \text{ct}_1 \leftarrow \mathcal{E}.\text{Enc}(\text{pk}_1, x; r_1) \) and \( \text{ct}_2 \leftarrow \mathcal{E}.\text{Enc}(\text{pk}_2, x; r_2) \) and \( \pi_1 \) is a NIZK proof of Equation 1. The algorithm outputs ciphertext \( \text{Ct} \).

5. **Dec(Sk_C, \( \text{Ct} \))** on input **secret key** \( \text{Sk}_C = (\text{diO}_{\text{Trap}_{1,3}}, t, \text{Ct}') \) and ciphertext \( \text{Ct} \), the algorithm outputs \( \text{diO}_{\text{Trap}_{1,3}}[\text{Ct}, \text{Ct}'] \).
6. RecFake(Sk_C, Fk_C, Ct, x) on input secret key Sk_C = (diO_{Trap1,3}, t, Ct'), fake key Fk_C = z, where $t = f(z)$, ciphertext $Ct = (ct_1, ct_2, \pi_1)$ and message $x$, does the following: Compute $\hat{Ct} := (\hat{ct}_3, \hat{ct}_4, \hat{\pi}_2)$, where $\hat{ct}_3 \leftarrow E.Enc(pk_3, (z, Ct, x); r_3)$ and $\hat{ct}_4 \leftarrow E.Enc(pk_4, (z, Ct, x); r_4)$ and $\hat{\pi}_2$ is a NIZK proof of Equation 2. The new secret key for circuit $C$ is $Sk_C = (diO_{Trap1,3}, t, \hat{Ct})$.

Correctness follows immediately from the correctness of the diO, PKE, SSS-NIZK, and the description of the trapdoor circuits described below.

\[
\text{Tr}_{\pi_1,\pi_2}[C, crs, crs', sk_i, sk_j, f, t](Ct = (ct_1, ct_2, \pi_1), Ct' = (ct_3, ct_4, \pi_2))
\]

The algorithm does the following:
1. Check that $\pi_1$ is valid NIZK proof (using the NIZK.Verify algorithm and crs) for the NP-statement
   \[\exists x, r_1, r_2 : \]
   \[ct_1 = E.Enc(pk_1, x; r_1) \text{ and } ct_2 = E.Enc(pk_2, x; r_2)\]
   \[
   (1)
   \]
2. Check that $\pi_2$ is valid NIZK proof (using the NIZK.Verify algorithm and $crs'$) for the NP-statement
   \[\exists z, c, x, r_3, r_4 :
   \]
   \[ct_3 = E.Enc(pk_3, (z, c, x); r_3) \text{ and } ct_4 = E.Enc(pk_4, (z, c, x); r_4) \text{ and } f(z) = t
   \]
   \[
   (2)
   \]
3. If any check fails output 0.
4. $(z', c', x') \leftarrow E.Dec(sk_j, ct_j)$
5. If $c' = Ct$ then output $C(x')$; otherwise output $C(E.Dec(sk_i, ct_i))$.

\[\text{Remark 4.2. To allow a secret key to support faking against } n_c \text{ ciphertexts, like it is done in the receiver deniable scheme of the conference version, we attach to a secret key } n_c \text{ ciphertexts } Ct'_i, each being the double encryption of } (z_i, 0^a, 0^\lambda) \text{ for different } z_i \text{'s. Then, } \text{Tr}_{\pi_1,\pi_2} \text{ will contain the images under the one-way function } f \text{ of all } z_i \text{'s.}
\]

4.0.2. Proof of Security.: We state the following theorem.

\[\text{Theorem 4.3. If } diO \text{ is a differing-input obfuscator, } E \text{ is IND-CPA and } f \text{ is a one-way function then } MDCRecDenFE \text{ is a } (1, 1)-\text{multi-distributional receiver deniable in the sense of Definition 2.6.}
\]

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 PDFE = (PDFE.Setup, PDFE.Enc, PDFE.KeyGen, PDFE.Eval, 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 DenFE = (Setup, KeyGen, Enc, Dec, SendFake, RecFake) for Circuit as follows:

- Setup\((1^\lambda, 1^n)\) runs PDFE.Setup\((1^\lambda, 1^{n+\lambda})\) to get the pair (PDFE.Mpk, PDFE.Msk). Then, the master public key is Mpk = PDFE.Mpk and the master secret key is Msk = PDFE.Msk. The algorithm returns the pair \((\text{Mpk}, \text{Msk})\).
- Enc(\text{Mpk}, x) on input master public key Mpk = 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

\[
\text{Ct} = \text{PDFE.Enc}(\text{PDFE.Mpk}, x') .
\]

- KeyGen(\text{Msk}, C) on input master secret key Msk = 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

\[
\text{PDFE.Sk}_{C} = \text{PDFE.KeyGen}(\text{PDFE.Msk}, \text{Trap}[C, \mathcal{F}^{t,z,t',z'}]),
\]

where \( t = (t_1, \ldots, t_{n_c}), z = (z_1, \ldots, z_{n_c}), t' = (t'_1, \ldots, t'_{n_c}), z' = (z'_1, \ldots, z'_{n_c}) \) and \( \text{Trap}[C, \mathcal{F}^{t,z,t',z'}](x, s) \) 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 \( \text{Sk}_{C} = (t, z, t', z', \text{PDFE.Sk}_{C}) \).
- Eval(\text{Mpk}, Ct, \text{Sk}_{C}) on input master public key Mpk = PDFE.Mpk, Ct and secret key \( \text{Sk}_{C} = (t, z, t', z', \text{PDFE.Sk}_{C}) \) for circuit C, returns the output of PDFE.Eval(\text{PDFE.Mpk}, Ct, PDFE.Sk_{C}).
- SendFake(\text{Mpk}, Ct, x) on input master public key Mpk = 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.
- RecFake(\text{Msk}, C, Ct, x) on input the master secret key Msk = PDFE.Msk, a Boolean circuit C on \( n \)-bits input and 1-bit output, at most \( n_c \) ciphertexts \( \text{Ct} = (\text{Ct}_1, \ldots, \text{Ct}_\ell) \), for \( \ell \leq n_c \), and messages \( x = (x_1, \ldots, x_\ell) \), extracts \( s_i \) from each ciphertext \( \text{Ct}_i \) by using 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 cases:

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

Finally, RecFake computes \( \text{PDFE.Sk}_{C} = \text{PDFE.KeyGen}(\text{PDFE.Msk}, \text{Trap}[C, \mathcal{F}^{t,z,t',z'}]) \), and returns secret key \( \text{Sk}_{C} = (t, z, t', z', \text{PDFE.Sk}_{C}) \).
**Correctness** of our DenFE scheme follows from the correctness of the PDFE scheme and from the observation that, for randomly chosen \( t, z, t', z' \) and \( s \) and for all \( x \), \( \text{Trap}(C, F)^{t,z,t',z'}(x, s) = C(x) \) except with negligible probability.

### 5.2. Proof of Security

The proof of security is similar 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 \((\text{poly}, 1, \text{poly})\)-IND-Secure then DenFE is a \((n_c, \text{poly})\)-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.

\[
\text{Sim.RecFake}^{\text{PDFE.KeyGen}(\text{PDFE.Msk}, \cdot)}(C, x, s)
\]

The algorithm takes in input a circuit \( C \), messages \( x = (x_1, \ldots, x_\ell) \), strings \( s = (s_1, \ldots, s_\ell) \) 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 \( \{0, 1\}^{|\lambda|} \) and distinguishes between the following two case:

- If \( C(x_i) = 1 \), it sets \( z_i = f_{s_i}(t_i) \) and chooses random \( z_i' \in \{0, 1\}^{|\lambda|} \).
- If \( C(x_i) = 0 \), it sets \( z_i' = f_{s_i}(t_i') \) and chooses random \( z_i \in \{0, 1\}^{|\lambda|} \).

Finally, the algorithm computes \( \text{PDFE.Ksk}_C = \text{PDFE.KeyGen}(\text{PDFE.Msk}, \text{Trap}(C, F)^{t,z,t',z'}) \), and returns secret key \( \text{Ksk}_C = (t, z, t', z', \text{PDFE.Ksk}_C) \).

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 \( \text{RealDenExp} \) where the sender-coerce oracle, let us call it \( \mathcal{E}_1^* \), and the receiver-coerce oracle, let us call it \( \mathcal{K}_1^* \), are the following:

\[
\begin{align*}
\mathcal{E}_1^*(x, y) & \quad \mathcal{K}_1^*(C, Ct, x) \\
(s_i \leftarrow \{0, 1\}^\lambda)_{i \in [n_c]} & \quad \text{Sk}_C \leftarrow \text{KeyGen}(\text{Msk}, C); \\
(Ct_i \leftarrow \text{PDFE.Enc}(\text{Mpk}, (x_i, s_i); r_i))_{i \in [n_c]} & \\
\text{Output: } ((\text{Ct}_i), (s_i, r_i)) & \\
\end{align*}
\]

Notice that \( \mathcal{E}_1^* \) 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 \( \mathcal{K}_1^* = \mathcal{K}_1 \).

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:

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

\[
\mathcal{E}_3^*(x, y) = \begin{cases} 
(s_i \leftarrow \{0, 1\}^\lambda)_{i \in [n_c]} \\
(s'_i \leftarrow \{0, 1\}^\lambda)_{i \in [n_c]} \\
(C_t_i \leftarrow \text{PDFE.Enc}(\text{Mpk}, (x_i, s_i); r_i))_{i \in [n_c]} \\
\text{Output: } ((C_t_i), (s_i, r_i)) 
\end{cases}
\]

\[
\mathcal{K}_3^*(C, x, y) = \begin{cases} 
\text{Sk}_C \leftarrow \text{Sim.RecFake}^{\text{PDFE.KeyGen(\text{PDFE.Msk}.)}}(C', x, s') 
\end{cases}
\]

where $s' = (s'_1, \ldots, s'_{n_c})$ is the randomness sampled by $\mathcal{E}_3^*$.

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

\[
\mathcal{E}_4^*(x, y) = \begin{cases} 
(s_i \leftarrow \{0, 1\}^\lambda)_{i \in [n_c]} \\
(s'_i \leftarrow \{0, 1\}^\lambda)_{i \in [n_c]} \\
(C_t_i \leftarrow \text{PDFE.Enc}(\text{Mpk}, (y_i, s'_i))_{i \in [n_c]} \\
\text{Output: } ((C_t_i), (x_i, s_i)) 
\end{cases}
\]

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

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 $1 \leq j \leq n_c + 1$:** This is the same as $H_2$ except that the following new sender-coerce oracle is used:

\[
\mathcal{E}_{2,j}^*(x, y) = \begin{cases} 
(s_i \leftarrow \{0, 1\}^\lambda)_{i \in [n_c]} \\
(s'_i \leftarrow \{0, 1\}^\lambda)_{i \in [n_c]} \\
(C_t_i \leftarrow \text{PDFE.Enc}(\text{Mpk}, (x_i, s_i); r_i) \\
\text{Then, for } i < j, e_i \leftarrow \text{PDFE.Explain}(C_t_i, (x_i, s_i))_{i \in [n_c]}, \\
\text{Output: } ((C_t_i), ((s_1, e_1), \ldots, (s_{j-1}, e_{j-1}), (s_j, r_j), \ldots, (s_{n_c}, r_{n_c}))) 
\end{cases}
\]

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 $D$ and adversary $A = (A_0, A_1)$ for which $H_{2,1}$ and $H_{2,k+1}$ are not computationally indistinguishable. Then $A$ and $D$ can be used to construct a successful IND-EXPL adversary $B$ for PDFE. Specifically, $B = (B_0, B_1)$ does the following.

- $B_0$ on input PDFE master public key $Mpk$, and master secret key of the PDFE’s underlying functional encryption $FE.Msk$, runs $A_0$ on input master public key $Mpk$ and answers $A_0$’s queries to $O_1$ and $O_2$ by using $Mpk$ and $FE.Msk$.

  Eventually, $A_0$ outputs $x^* = (x^*_1, \ldots, x^*_n)$, $y^* = (y^*_1, \ldots, y^*_n)$ and its state $st$. Then, $B_0$ chooses random $s \in \{0, 1\}^n$ and outputs $(x^*_k, s^*)$, and put in its state the state of $A_0$ and its entire computation.

- $B_1$ on input ciphertext $Ct^*$, encryption of $(x^*_k, s^*)$, randomness $r^*$, which is the real randomness used to generate $Ct^*$ or the randomness obtained by the explain algorithm, and state $st$, does the following: $B_1$, chooses $(s_i \leftarrow \{0, 1\}^\lambda)_{i \in [n]}$ and $(s'_i \leftarrow \{0, 1\}^\lambda)_{i \in [n]}$, then, for $i \in [n] \setminus \{k\}$, sets $Ct^*_i = Encrypt(Mpk, (x^*_i, s_i))$, and for index $k$, $B_1$ sets $Ct^*_k = Ct^*$. Finally, for $i < j$, $B_1$ sets $e_i \leftarrow PDFE.Explain(Ct_i, (x_i, s_i))_{i \in [n]}$.

  Finally, $B_1$ runs $A_1$ on input challenge ciphertexts and sender randomness $Ct^*_1, \ldots, Ct^*_n$ and sender randomness $((s_1, e_1), \ldots, (s_{j-1}, e_{j-1}), (s_j, r_j), \ldots, (s_n, r_n))$, and answers $A_1$’s queries to $O_1$ and $O_2$ and to the receiver-coerce oracle $K$, by using $Mpk$ and $FE.Msk$.

  Eventually, $A_1$ returns its output and $B_1$ passes it to the distinguisher $D$ and returns $D$’s output as its own output.

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

**Indistinguishability of $H_3$ and $H_2$:** 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. 

\[\square\]

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


