The Edited Truth

Shafi Goldwasser[*] Saleet Klein[†] Daniel Wichs[‡]

July 19, 2017

Abstract

We introduce two new cryptographic notions in the realm of public and symmetric key encryption.

- *Encryption with invisible edits* is an encryption scheme with two tiers of users: "privileged" and "unprivileged". Privileged users know a key pair $(\mathsf{pk}, \mathsf{sk})$ and "unprivileged" users know a key pair $(\mathsf{pk}_e, \mathsf{sk}_e)$ which is associated with an underlying edit e to be applied to messages encrypted. Each key pair on its own works exactly as in standard public-key encryption, but when an unprivileged user attempts to decrypt a ciphertext generated by a privileged user of an underlying plaintext m, it will be decrypted to an edited $m' = \mathsf{Edit}(m, e)$. Here, Edit is some supported edit function and e is a description of the particular edit to be applied. For example, we might want the edit to overwrite several sensitive blocks of data, replace all occurrences of one word with a different word, airbrush an encrypted image, etc. A user shouldn't be able to tell whether he's an unprivileged or a privileged user.
- An *encryption with deniable edits* is an encryption scheme which allows a user who owns a ciphertext c encrypting a large corpus of data m under a secret key sk, to generate an alternative but legitimate looking secret key $\mathsf{sk}_{c,e}$ that decrypts c to an "edited" version of the data $m' = \mathsf{Edit}(m, e)$. This generalizes classical receiver deniable encryption, which can be viewed as a special case of deniable edits where the edit function performs a complete replacement of the original data. The new flexibility allows us to design solutions with much smaller key sizes than required in classical receiver deniable encryption, and in particular allows the key size to only scale with the description size of the edit e which can be much smaller than the size of the plaintext data m.

We construct encryption schemes with deniable and invisible edits for any polynomial-time computable edit function under minimal assumptions: in the public-key setting we only require the existence of standard public-key encryption and in the symmetric-key setting we only require the existence of one-way functions.

The solutions to both problems use common ideas, however there is a significant conceptual difference between deniable edits and invisible edits. Whereas encryption with deniable edits enables a user to modify the meaning of a single ciphertext in hindsight, the goal of encryption with invisible edits is to enable ongoing modifications of multiple ciphertexts.

[*] MIT, EECS and Weizmann Institute, Department of Computer Science and Applied Mathematics. shafi@csail.mit.edu. Research supported by NSF MACS - CNS-1413920 ,DARPA IBM - W911NF-15-C-0236, SIMONS Investigator award Agreement Dated 6-5-12

[†] MIT, EECS. saleet@csail.mit.edu. Research supported by an Akamai Presidential Fellowship and NSF MACS - CNS-1413920 ,DARPA IBM - W911NF-15-C-0236, SIMONS Investigator award Agreement Dated 6-5-12

[‡] Northeastern University, Department of Computer Science. wichs@ccs.neu.edu. Research supported by NSF grants CNS-1314722, CNS-1413964.

1 Introduction

In this paper, we introduce two novel cryptographic notions in the realm of public and symmetric key encryption: *Encryption with invisible edits (IEdit)* and *Encryption with deniable edits (DEdit)*.

We construct both asymmetric and symmetric key versions of IEdit and DEdit schemes, under minimal assumptions using the machinery of garbled circuits. In particular, we can get such schemes in the public-key setting using only public key encryption and in the symmetric-key setting using only one-way functions. Our constructions rely on a simple but delicate use of functional encryption (FE), further illustrating the incredible versatility of this powerful abstraction.

We proceed to describe the new notions and our constructions.

1.1 Invisible Edits

Alice is a company boss and her secretary Bob is in charge of going through her e-mail (which is naturally all encrypted) and responding to routine requests. However, sometimes other bosses will send e-mails containing information that Bob should not see, for example discussing layoffs among the secretarial staff. Alice would like to give Bob a secret key which will invisibly introduce some careful edits to all such e-mails (e.g., replaces the word "layoffs" with "bonuses"), even ones sent in the future. Ideally, Bob should not know anything about what edits are being introduced and should even be oblivious to the fact that he does not have Alice's real secret key which decrypts all e-mails correctly.

Encryption with Invisible Edits. To solve the above problem, we introduce a new cryptographic primitive that we call encryption with *invisible edits* (IEdit). IEdit is an encryption system which allows dispensing computationally indistinguishable decryption keys which each decrypt a ciphertext to a different "edited" plaintexts. A user cannot tell whether or not his decryption key is introducing edits.

In more detail, such a scheme allows us to create "privileged" encryption/decryption key pairs $(\mathsf{pk}, \mathsf{sk})$ and "unprivileged" encryption/decryption key pairs $(\mathsf{pk}_e, \mathsf{sk}_e)$ tied to some edit e. Both key pairs individually work correctly, meaning that a message encrypted under pk (resp. pk_e) will decrypt correctly under sk (resp. sk_e). However, when a privileged user encrypts some message m under pk, the unprivileged user will decrypt it to $m' = \mathsf{Edit}(m, e)$ under sk_e. Here, we think of Edit as some edit function which is specified as part of the scheme and e is the description of the particular edit that should be applied. For example, we might consider an edit function that performs a small number of insertions and deletions on blocks of the data, as specified by e. Alternatively, the edit function could be a complex suite of image-editing tools and e could specify a series of transformations (e.g., crop, rotate, blur, airbrush, etc.) to be performed on the encrypted image. More generally, we can think of the edit e as a Turing Machine and the edit function as a universal Turing Machine which runs $e(m)$.

A user shouldn't be able to tell whether he is privileged or unprivileged. In particular, the user can't tell whether he's an unprivileged user that has $(\mathsf{pk}_e, \mathsf{sk}_e)$ and is receiving ciphertexts from privileged users that are encrypting some messages m_i under pk while he is decrypting the edited versions $m'_i = \mathsf{Edit}(m_i, e)$, or whether he is a privileged user that gets $(\mathsf{pk}, \mathsf{sk})$ and is receiving ciphertexts from other privileged users that are really encrypting m'_i under pk.

In addition to considering the problem of invisible edits in the public-key setting, we also consider a symmetric-key variant of the problem where the key sk (resp. sk_e) is used for both encryption and decryption. In the symmetric-key case, we consider two potential variants.

Dual-Key Variant. In the dual key variant, the privileged/unprivileged keys sk and sk_e look indistinguishable and a user cannot tell which key he has.

Dual-Scheme Variant. In the dual scheme variant, the privileged and unprivileged users have completely different keys and even encryption/decryption procedures. Therefore users can tell whether they are privileged or unprivileged. However, unprivileged users still cannot tell whether their key always decrypts all ciphertexts correctly or whether it is introducing edits to data encrypted by privileged users.

Intuitively, the dual-key variant is more desirable.

Invisible Edits: Our Results. We construct encryption with invisible edits in the public-key setting, under the minimal assumption that public-key encryption exists. In the symmetric-key setting, we construct the weaker dual-scheme variant under one-way functions but leave it as an interesting open problem to also construct the stronger dual-key variant under one-way functions or show that it requires public key encryption.

The secret key (and public key) size of our schemes is linear in the edit description size $|e|$. The run-time of the encryption/decryption procedures and the ciphertext size are linear in the circuit size of the edit function. In the public-key setting, we can use identity based encryption (IBE) to further reduce the public-key size to only depend on the security parameter.

1.2 Deniable Edits

DEdit is a different but technically related notion to IEdit, which extends the classical notion of receiver deniable encryption [CDNO97] to allow the legal owner (and originator) of a secret key to produce an alternative computationally indistinguishable secret key under which a targeted ciphertext decrypts to an "edited" plaintext. The description size of the edits to be applied to the original plaintext can be much smaller than the size of the plaintext itself. This will allow us to design solutions, where the secret key size is only proportional to the description size of the edit, but can be much smaller than the message size.

As a motivating scenario, consider Alice who is an owner of a private server hosting a large corpus of data which is encrypted under a small secret key held by Alice on a separate device. Circumstances cause Alice to become the subject of scrutiny, the server is seized by investigators, and Alice must hand over her secret key. Although most of the data is innocuous, the server might contain a few private photos, confidential recommendation letters, etc. Alice wants to comply, but give a different secret key which looks legitimate but decrypts the data to a "lightly edited" version where the sensitive content is appropriately modified. Typically, the description of the edits to be broadly applied can be succinctly summarized and is much smaller than the size of the data.

New Primitive: Encrytption with Deniable Edits. To solve the above problem, we introduce a new cryptographic primitive that we call encryption with *deniable edits*. Such a scheme can be used to encrypt a potentially huge message m using a relatively short secret key sk to derive a ciphertext c. Later, it should be possible to come up with a legitimate looking secret key $\mathsf{sk}_{c,e}$ that decrypts the ciphertext c to an edited message $m' = \mathsf{Edit}(m, e)$, where Edit is some "edit function" specified by the scheme and e is a description of the particular edit that should be applied. We envision that the description-size of the edit $|e|$ is relatively small, and much smaller than the potentially huge message size $|m|$. Therefore, although we necessarily need to allow the secret key size $|\mathsf{sk}|$ to grow with the edit description size $|e|$, we do not want it to depend on the message size $|m|$. The exact same notion can be defined in either public or symmetric key settings.

Relation to Deniable Encryption and its Limitations. One can think of encryption with deniable edits as a more flexible version of *receiver deniable encryption*, introduced by Canetti, Dwork, Naor and Ostrovsky [CDNO97]. In receiver deniable encryption, it is possible to come up with a secret key $\mathsf{sk}_{c,m'}$ that decrypts a ciphertext c to an arbitrary message m'. However, the size of the secret key in deniable encryption schemes must necessarily be at least as large as the message size. This makes such schemes unsuitable for encrypting a large messages such as the entire hard-disk contents. Encryption with deniable edits provides flexibility by allowing the secret key size to only scale with the edit description size which can potentially be much smaller than the message size. Naturally, we can recover the notion of receiver deniable encryption as a special case by taking the edit function $\mathsf{Edit}(m, e) = e$ which simply overwrites the encrypted message m with the value e, of size $|e| = |m|$. We discuss the relevant literature on deniable encryption and its relation to our work in Section 1.5.

Since encryption with deniable edits generalizes receiver deniable encryption, it also *inherits its limitations.* In particular, Bedlin et al. [BNNO11] show that the most natural definition of deniability, where the original secret key sk can be used to create a legitimate-looking sk' which is indistinguishable from sk but decrypts a selected ciphertext c differently, cannot be achieved. Instead, we consider two potential ways to weaken the definition:

Dual-Key Variant. The key-generation algorithm outputs a secret decryption key sk along with a secret denying key dk. Most users can immediately discard dk since it is not needed for decryption. However, users that keep dk (e.g., hidden in their basement) can use it to later produce a modified secret key $\mathsf{sk}_{c,e}$ which looks legitimate but decrypts a selected ciphertext c to an edited message.

Dual-Scheme Variant. There are two entirely different encryption schemes: a "default" scheme and a "denying" scheme. Most users are expected to use the default scheme. However, if a user instead uses the denying scheme, she can take her secret key sk and a ciphertext c and produce a secret key $\mathsf{sk}_{c,e}$ which makes it look as though she was using the default scheme but c decrypts to an edited message.[1]

Intuitively, one can think of the dual-key variant as a special case of the dual-scheme variant, where the default and denying schemes are essentially identical, except that in the latter the user keeps both $(\mathsf{sk}, \mathsf{dk})$ while in the former she only keeps sk. Therefore, we view the dual-key variant as more desirable. In the public-key setting, it turns out that the two variants are essentially identical and therefore we can only consider the more compelling dual-key variant. However, we do not know if equivalence holds in the symmetric-key setting and therefore consider both variants there.

Deniable Edits: Our Results. We construct encryption with deniable edits for arbitrary polynomial-time edit functions under essentially minimal assumptions. In the public-key setting, we construct such a scheme from any standard public-key encryption. In the symmetric-key setting, we show how to construct the dual-scheme variant under the minimal assumption that one-way functions exist. However, we leave it as an interesting open problem whether one can also construct the stronger dual-key variant under one-way functions or whether it requires public key encryption.

The secret key (and public key) size of our schemes is linear in the edit description size $|e|$. The run-time of the encryption/decryption procedures and the ciphertext size are linear in the circuit size of the edit function. In the public-key setting, we can use identity based encryption (IBE) to further reduce the public-key size to only depend on the security parameter.

We also discuss an extension of our schemes to deniably editing some bounded number of ciphertexts (rather than just one) at the cost of having the secret key size scale with this bound. Furthermore we show how to extend our schemes to be able to deniably produce not just a secret key but also the randomness of the key generation algorithm (see Section 4.4).

1.3 Comparison: Deniable Edits, Invisible Edits and Functional Encryption

It is useful to compare the notions of deniable edits, invisible edits and functional encryption. For concreteness, we consider the comparison in the public-key setting. In all three cases, we can produce a secret key tied to some edit e and ensure that it decrypts encrypted data m to some modified value $\mathsf{Edit}(m, e)$. However, there are crucial differences between the three primitives.

- In functional encryption, we are not hiding the fact that the secret key sk_e is introducing edits to the encrypted data. In fact, a user that has the (master) public key pk will be immediately aware of the fact that when he encrypts a message m via pk and decrypts via sk_e he gets an edited value $m' = \mathsf{Edit}(m, e)$. This is in contrast to both encryption with deniable and invisible edits, where we do want to hide the fact that edits are being introduced.

- In encryption with deniable edits, we create a secret key $\mathsf{sk}_{c,e}$ which only introduces edits to the decryption of a single specified ciphertext c. Therefore, even if a user has pk and can create his own ciphertexts, he will not observe any edits being introduced.

- In encryption with invisible edits, we hide the fact that the secret key sk_e is introducing edits by also creating a matching public key pk_e. Encryptions under pk_e decrypt correctly (with no edits) under sk_e and therefore a user that has $(\mathsf{pk}_e, \mathsf{sk}_e)$ cannot tell that edits are being introduced. However, if other users encrypt data under pk, it will consistently decrypt to an edited version under sk_e.

[1] This variant was also called multi-distributional deniable encryption in recent works.

Despite the major differences between the three primitives, we will use functional encryption (based on garbled circuits) as a tool to get relatively simple constructions of the other two primitives.

We can think of using a scheme with invisible edits, which targets multiple ciphertexts, in scenarios involving deniability. In particular, consider the case where Alice is running an e-mail server storing a large corpus of individually encrypted e-mails $c_1 = \mathsf{Enc}_{\mathsf{pk}}(m_1), \ldots, c_T = \mathsf{Enc}_{\mathsf{pk}}(m_T)$. She comes under an investigation and wants to give a secret key that applies some simple edit across *all* the e-mails (e.g., replaces one word with a different word). Using an encryption scheme with deniable edits this would only be possible if all of the e-mails were encrypted simultaneously in one ciphertext, but that's not the case here. Using encryption with invisible edits, we can solve the problem at the cost of Alice having to be able to convincingly hand over to the investigators not only her modified secret key (giving sk_e instead of sk) but also her modified encryption key (giving pk_e instead of pk). This makes sense in the symmetric-key setting if we think of the encryption key pk as also being private or even in scenarios where Alice gives her encryption key pk to a small circle of semi-trusted parties but does not publish it widely.

1.4 Our Techniques

All of our constructions rely on simple but delicate use of functional encryption (FE), further illustrating the versatility of this powerful abstraction. A public-key FE scheme for some function $F(x, y)$ comes with a master public key mpk that can be used to generate ciphertexts $c \leftarrow \mathsf{Enc}_{\mathsf{mpk}}(x)$ encrypting some values x, and a master secret key msk that can be used to generate secret keys $\mathsf{sk}_y \leftarrow \mathsf{Gen}_{\mathsf{msk}}(y)$ associated with values y. When we decrypt the ciphertext c with the secret key sk_y we get $\mathsf{Dec}_{\mathsf{sk}_y}(c) = F(x, y)$. We only need FE schemes that are secure in the setting where the adversary sees a single secret key, which we know how to construct under minimal assumptions using the machinery of garbled circuits. In particular, we can get such schemes in the public-key setting using only public key encryption and in the symmetric-key setting using only one-way functions by the work of Sahai and Seyalioglu [SS10]

Invisible Edits. Let us start with our construction of public-key encryption with invisible edits, for some edit function $\mathsf{Edit}(m, e)$.

As an initial idea, we might consider taking a functional encryption scheme for the function $F(m, e) = \mathsf{Edit}(m, e)$ where ciphertexts encrypt messages m and secret keys are associated with edits e, and set the priviledged secret key $\mathsf{sk}_{\mathsf{id}}$ to be a secret key for the identity edit id such that $\mathsf{Edit}(m, \mathsf{id}) = m$, whereas unpriviledged secret key pair would be sk_e such that $\mathsf{Dec}_{\mathsf{sk}_e}(c) = \mathsf{Edit}(m, e)$. Unfortunately, this initial idea does not work since it's easy to distinguish sk_e from $\mathsf{sk}_{\mathsf{id}}$ by generating encryptions of known plaintexts and seeing how they decrypt.

To fix the above idea, we take a functional encryption scheme for a more complicated function $F(x, y)$ which interprets $x = (m, k)$ and tests if $y \oplus k$ is of the form $0^\lambda || e$ where λ is the security parameter; if so it outputs $\mathsf{Edit}(m, e)$ and else it outputs m. A "privileged" key pair consists of a public key (mpk, k) and secret key sk_y where k, y are random and independent. To encrypt a message m, we use the FE scheme to encrypt the tuple $x = (m, k)$ where k comes from the public key. An "unprivileged" key pair consists of a public key (mpk, k') and a secret key $\mathsf{sk}_{y'}$ where k' is random and $y' = (0^\lambda || e) \oplus k$.

Notice that the privileged and unprivileged key pairs are individually identically distributed, but there is a correlation between them. If we encrypt a message m with a privileged (resp. unprivileged) public-key and then decrypt the resulting ciphertext with a privileged (resp. unprivileged) secret key than since k, y (resp. k', y') are random and independent we decrypt the correct value $F(x, y) = m$ with all but negligible probability. However, if we encrypt a message m with a privileged public key which corresponds to an FE encryption of $x = (m, k)$ and then decrypt with an unpriveleged secret key $\mathsf{sk}_{y'}$ then we get $F(x, y') = \mathsf{Edit}(m, e)$.

We argue that one cannot distinguish between having a privileged key pair $((\mathsf{mpk}, k), \mathsf{sk}_y)$ and seeing privileged encryptions of $m' = \mathsf{Edit}(m, e)$ which corresponds to FE encryptions of $x' = (m', k)$, versus having an unprivileged key pair $((\mathsf{mpk}, k'), \mathsf{sk}_{y'})$ and seeing privileged encryptions of m which corresponds to FE encryptions of $x = (m, k)$. In particular, since the key pairs are identically distributed, the only difference between these games is the conditional distribution of x versus x', but since $F(x', y) = F(x, y') = m'$, this difference is hidden by FE security.

Our solution for symmetric-key encryption with invisible edits is again analogous, but relying on symmetric-key FE instead of public-key FE.

Deniable Edits. As an initial idea, we might consider taking a functional encryption scheme for the function $F(m, e) = \mathsf{Edit}(m, e)$ where ciphertexts encrypt messages m and secret keys are associated with edits e. We set the public-key of our scheme to be the FE master public-key mpk and the secret key $\mathsf{sk}_{\mathsf{id}}$ would be a secret key for the identity edit id such that $\mathsf{Edit}(m, \mathsf{id}) = m$. A user that wants to be able to deny in the future would also keep a "denying key" dk which we set to be the FE master secret key $\mathsf{dk} = \mathsf{msk}$. To later claim that some ciphertext c encrypting a message m is really an encryption of $m' = \mathsf{Edit}(m, e)$ the user would use $\mathsf{dk} = \mathsf{msk}$ to generate the secret key sk_e for the edit e. Unfortunately, this initial idea does not work since it's easy to distinguish sk_e from $\mathsf{sk}_{\mathsf{id}}$ by generating encryptions of known plaintexts and seeing how they decrypt. What we really need is for the denying procedure to output a secret key that only edits the value in one particular targeted ciphertext c, but otherwise decrypts all other ciphertexts correctly.

To fix the above, we use a similar idea as in the case of invisible edits. We take a functional encryption scheme for a more complicated function $F(x, y)$ which interprets $x = (m, k)$ and tests if $y \oplus k$ is of the form $0^\lambda || e$ where λ is the security parameter; if so it outputs $\mathsf{Edit}(m, e)$ and else it outputs m. We set the public-key of our encryption scheme to be mpk, the secret key to be sk_y for a uniformly random value y, and the denying key to be $\mathsf{dk} = \mathsf{msk}$. To encrypt a message m, the encryption procedure chooses a fresh value k on each invocation (this is in contrast to the invisible edits construction where k was part of the public key) and uses the FE scheme to encrypt the tuple $x = (m, k)$ resulting in some ciphertext c. Notice that, since k, y are random and independent, $y \oplus k$ is not of the form $0^\lambda || e$ except with negligible probability and therefore decrypting the ciphertext c with the key sk_y results in the correct value $F(x, y) = m$. If the user wants to later claim that this particular ciphertext c is really an encryption of $m' = \mathsf{Edit}(m, e)$, she would use $\mathsf{dk} = \mathsf{msk}$ to generate a secret key $\mathsf{sk}_{y'}$ for the value $y' = (0^\lambda || e) \oplus k$ which decrypts c to $F(x, y') = \mathsf{Edit}(m, e)$. Notice that the original key sk_y and the new key $\mathsf{sk}_{y'}$ are identically distributed. We claim that one cannot distinguish between seeing $(c, \mathsf{sk}_{y'})$ and (c', sk_y) where c' is an actual encryption of $m' = \mathsf{Edit}(m, e)$, meaning that it is an FE encryption of $x' = (m', k')$ for a uniform k'. Since y and y' are individually identically distributed, the only difference between these tuples is the conditional distribution of x vs. x', but since $F(x', y) = F(x, y') = m'$, this difference is hidden by FE security.

Our solution for symmetric-key encryption with deniable edits is analogous, but relying on symmetric-key FE instead of public-key FE.

1.5 Related Work

The notion of *deniable encryption* was introduced by Canetti, Dwork, Naor and Ostrovsky [CDNO97]. They considered two separate facets of this notion: *sender deniability* considers the scenario where the encryptor is coerced to produce the random coins of the encryption algorithm, whereas *receiver deniability* considers the scenario where the decryptor is coerced to produce the secret key (or even the random coins of the key generation algorithm). As noted in several prior works, it is easy to protect against sender coercion by simply having senders erase the randomness they use after each encryption operation, and similarly the receiver can erase the randomness of the key generation algorithm. However, the receiver needs to keep her secret key for the long term in order to decrypt. Therefore, we view receiver deniability, where the receiver is coerced to produce her secret key, as the most important deniability scenario and focus on this in our work. Nevertheless, we mention that the other notions of deniability are also interesting and meaningful in settings where erasure is not technically or legally feasible.

Canetti et al. [CDNO97] construct both sender and receiver deniable public-key encryption schemes where it is possible to distinguish between the real key/randomness and the fake key/randomness with an inverse polynomial advantage. The work of Sahai and Waters [SW14] constructs a sender deniable with negligible distinguishing advantage using indisitnguishability obfuscation. Bedlin et al. [BNNO11] show that a negligible distinguishing advantage cannot be achieved for receiver deniability if we consider the most natural notion where, given a secret key sk and a honestly generated ciphertext c encrypting some message m, it is possible to generate a secret key $\mathsf{sk}_{c,m'}$ that decrypts c to an arbitrarily different message m'. Although they show this for public-key encryption, the result also naturally extends to CPA secure symmetric-key encryption (but not for one-time encryption, where the one-time pad is optimally deniable).

As we discussed, it is possible to circumvent the results of Bedlin et al. [BNNO11] by relaxing the notion of receiver deniability and considering dual-key or dual-scheme (also called multi-distributional) variants. The work of O'Neill, Peikert and Waters [OPW11] constructs a dual-scheme deniable public-key encryption which is simultaneously sender and receiver deniable (bi-deniable). The work of [DIO16] construct both dual-scheme and dual-key variants of receiver deniable *functional encryption*. Whereas in that work, functionality and deniability were orthogonal properties (i.e., the goal was to get a scheme which is simultaneously a functional encryption scheme and deniable), one can see our notion of deniable edits as a type of functional-deniability where the fake secret key invisibly applies a function to the encrypted message.

Deniable encryption is also very related to the concept of non-committing encryption [CFGN96, DN00, CDMW09]. On a very high level, the latter notion only requires the ability to equivocate ciphertexts that were specially generated by a simulator whereas the former notion requires the ability to equivocate honestly generated ciphertexts.

In both receiver-deniable and non-committing encryption, the secret key size is necessarily at least as large as the message size [Nie02]. This is because for every possible message m', there has to be a different secret key $\mathsf{sk}_{c,m'}$ that decrypts the given ciphertext c to m'. Our work *flexibly circumvents this lower bound in the setting of deniable encryption* by restricting the set of messages m' to which we can open the ciphertext to only be values of the type $m' = \mathsf{Edit}(m, e)$ where m is the message that was originally encrypted and e is the description of an edit. This allows the secret key size to only scale with the edit description size $|e|$ instead of the message size $|m|$.

The idea of restricting the set of messages to which a ciphertext can be opened in order to reduce the secret key size has been considered in several other prior works, both in the context of deniability and non-committing encryption. For example, the notions of *plan-ahead deniability* [CDNO97, OPW11] and *somewhat-non committing encryption* [GWZ09] fix a small set of messages to which a ciphertext can be opened at encryption time. In *somewhere equivocal (non-committing) encryption* [HJO$^+$16] it is possible to modify a few blocks of the encrypted data. Perhaps the closest concept to our work is the notion of *functionally equivocal (non-committing) encryption* from the recent work of Canetti, Poburinnaya and Venkitasubramaniam [CPV16]. In that work, it's possible to open a simulated encryption to any message $m' = f(x)$ which is in the range of some function f, where f can be an expanding function and the secret key size is only proportional to $|x|$ rather than to $|m'|$. The main differences with our work on deniable edits are: (1) we study deniability rather than the non-committing setting, meaning that we start with a real ciphertext of some message m rather than a simulated ciphertext, (2) we want to open the ciphertext to an edited messages $m' = \mathsf{Edit}(m, e)$ that depends on the original value m rather than just an arbitrary value in the range of some fixed function f.

2 Preliminaries

We introduce several preliminaries including notation and definitions of functional encryption. See Appendix A for additional standard cryptographic definitions.

Notation. We denote by $[n]$ the set $\{1, \dots, n\}$. For a string $x \in \{0,1\}^*$ we denote by $x[i]$ the i-th bit of x. If X is a random variable, a probability distribution, or a randomized algorithm we let $x \leftarrow X$ denote the process of sampling x according to X. If $\mathcal{X}$ is a set, we let $x \leftarrow \mathcal{X}$ denote the process of sampling x uniformly at random from $\mathcal{X}$.

2.1 Single-Key Functional-Encryption

We now present definition of public and symmetric key functional encryption. We only require a weak notions of security where (1) the adversary only sees at most a single secret key and (2) the adversary has to selectively choose the secret key before it gets the challenge ciphertext.

Definition 2.1 (Single-Key PK FE). *A* single-key public-key functional-encryption scheme (PK FE) *for a function* $F : \{0,1\}^{n_1(\lambda)} \times \{0,1\}^{n_2(\lambda)} \to \{0,1\}^{n_3(\lambda)}$ *consists of PPT algorithms* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following syntax:*

- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *generates a master secret-key* msk *and master public key* mpk.
- $\mathsf{sk}_y \leftarrow \mathsf{Gen}_{\mathsf{msk}}(y)$ *takes an input* $y \in \{0,1\}^{n_2(\lambda)}$, *generates a secret-key* sk_y.
- $c \leftarrow \mathsf{Enc}_{\mathsf{mpk}}(x)$ *takes an input* $x \in \{0,1\}^{n_1(\lambda)}$, *outputs an encryption of* x.
- $F(x,y) = \mathsf{Dec}_{\mathsf{sk}_y}(c)$ *outputs* $F(x,y) \in \{0,1\}^{n_3(\lambda)}$.

The scheme should satisfy the following properties:

Correctness *For every security parameter* λ, *message* $x \in \{0,1\}^{n_1(\lambda)}$, *and* $y \in \{0,1\}^{n_2(\lambda)}$*:*

$$\Pr\left[F(x,y) = \mathsf{Dec}_{\mathsf{sk}_y}(\mathsf{Enc}_{\mathsf{mpk}}(x)) \;\middle|\; \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sk}_y \leftarrow \mathsf{Gen}_{\mathsf{msk}}(y) \end{array} \right] = 1.$$

Single-Key PK FE Security. *We define the "single-key public-key functional encryption game"* $\mathsf{FEGame}^b_{\mathcal{A}}(\lambda)$ *between an adversary* $\mathcal{A}$ *and a challenger with a challenge bit* $b \in \{0,1\}$ *as follows:*

- *Sample* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and send* mpk *to* $\mathcal{A}$.
- *The adversary* $\mathcal{A}$ *chooses* $y \in \{0,1\}^{n_2(\lambda)} \cup \{\perp\}$.
- *If* $y \neq \perp$, *sample* $\mathsf{sk}_y \leftarrow \mathsf{Gen}_{\mathsf{msk}}(y)$ *and send* sk_y *to* $\mathcal{A}$.
- *The adversary* $\mathcal{A}$ *chooses messages* $x_0, x_1 \in \{0,1\}^{n_1(\lambda)}$ *such that if* $y \neq \perp$ *then* $F(x_0, y) = F(x_1, y)$.
- *The adversary* $\mathcal{A}$ *gets a challenge* $\mathsf{Enc}_{\mathsf{mpk}}(x_b)$ *and eventually outputs a bit* b' *which we define as the output of the game.*

We require that for all PPT adversary $\mathcal{A}$ *we have*

$$|\Pr[\mathsf{FEGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{FEGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

Definition 2.2 (Single-Key SK FE)**.** *A* single-key symmetric-key functional-encryption scheme (SK FE) *for a function* $F : \{0,1\}^{n_1(\lambda)} \times \{0,1\}^{n_2(\lambda)} \to \{0,1\}^{n_3(\lambda)}$ *consists of PPT algorithms* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following syntax:*

- $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ *generates a master secret-key* msk.
- $\mathsf{sk}_y \leftarrow \mathsf{Gen}_{\mathsf{msk}}(y)$ *takes an input* $y \in \{0,1\}^{n_2(\lambda)}$, *generates a functional secret-key* sk_y.
- $c \leftarrow \mathsf{Enc}_{\mathsf{msk}}(x)$ *takes an input* $x \in \{0,1\}^{n_1(\lambda)}$, *outputs an encryption of* x
- $F(x,y) = \mathsf{Dec}_{\mathsf{sk}_y}(c)$ *outputs a message* $F(x,y) \in \{0,1\}^{n_3(\lambda)}$.

The scheme should satisfy the following properties:

Correctness *For every security parameter* λ, *message* $x \in \{0,1\}^{n_1(\lambda)}$, *and* $y \in \{0,1\}^{n_1(\lambda)}$*:*

$$\Pr\left[F(x,y) = \mathsf{Dec}_{\mathsf{sk}_y}(\mathsf{Enc}_{\mathsf{msk}}(x)) \;\middle|\; \begin{array}{l} \mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sk}_y \leftarrow \mathsf{Gen}_{\mathsf{msk}}(y) \end{array} \right] = 1.$$

Single-Key SK FE Security. *We define the "single-key secret-key functional encryption game"* $\mathsf{FEGame}^b_{\mathcal{A}}(\lambda)$ *between an adversary* $\mathcal{A}$ *and a challenger with a challenge bit* $b \in \{0,1\}$ *as follows:*

- *Sample* $\mathsf{msk} \leftarrow \mathsf{Setup}(\lambda)$ *and let* $\mathcal{O}(\cdot)$ *be an encryption oracle* $\mathcal{O}(\cdot) := \mathsf{Enc}_{\mathsf{msk}}(\cdot)$
- *The adversary gets access to the encryption oracle* $\mathcal{A}^{\mathcal{O}}$ *and eventually chooses* $y \in \{0,1\}^{n_2(\lambda)} \cup \{\perp\}$.
- *If* $y \neq \perp$, *sample* $\mathsf{sk}_y \leftarrow \mathsf{Gen}_{\mathsf{msk}}(y)$ *and send* sk_y *to* $\mathcal{A}$
- *The adversary* $\mathcal{A}^{\mathcal{O}}(\mathsf{sk}_y)$ *gets further access to the encryption oracle and eventually chooses messages* x_0, x_1 *such that if* $y \neq \perp$ *then* $F(x_0, y) = F(x_1, y)$.
- *The adversary* $\mathcal{A}^{\mathcal{O}}(\mathsf{sk}_y, c)$ *gets a challenge message* $c \leftarrow \mathsf{Enc}_{\mathsf{msk}}(x_b)$ *and further access to the encryption oracle, and eventually outputs a bit* b' *which we define as the output of the game.*

We require that for all PPT adversary $\mathcal{A}$ *we have*

$$|\Pr[\mathsf{FEGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{FEGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

Special Encryption/Decryption. We will require two additional properties from our FE schemes. Informally, a symmetric-key FE with a *special encryption* allows one to encrypt given a secret-key sk_y instead of msk while ensuring that the two methods are indistinguishable even given sk_y. A symmetric-key or public-key FE with *special decryption* allows one to decrypt with msk to recover the entire value x.

Definition 2.3 (Special Encryption). *We say that a symmetric-key functional encryption scheme* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *has a* special encryption *if the syntax of the* Enc *algorithm can be extended to work with a secret key* sk_y *instead of a master secret key* msk*, and for all PPT adversary* $\mathcal{A}$ *we have*

$$|\Pr[\mathsf{EncGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{EncGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

where $\mathsf{EncGame}^b_{\mathcal{A}}(\lambda)$ *is a game between an adversary and a challenger with a challenge bit* $b \in \{0,1\}$*, defined as follows:*

- *The adversary* $\mathcal{A}$ *chooses* $y \in \{0,1\}^{n_2(\lambda)}$
- *Sample* $\mathsf{msk} \leftarrow \mathsf{Setup}(\lambda)$ *and* $\mathsf{sk}_y \leftarrow \mathsf{Gen}_{\mathsf{msk}}(y)$*, and let* $\mathcal{O}(\cdot)$ *be an encryption oracle*

$$\mathcal{O}(\cdot) := \begin{cases} \mathsf{Enc}_{\mathsf{msk}}(\cdot) & b = 0 \\ \mathsf{Enc}_{\mathsf{sk}_y}(\cdot) & b = 1 \end{cases}$$

- *The adversary* $\mathcal{A}^{\mathcal{O}}(\mathsf{sk}_y)$ *gets access to the encryption oracle and the secret key, and eventually outputs a bit* b' *which we define as the output of the game.*

Definition 2.4 (Special Decryption). *We say that a symmetric-key functional encryption scheme* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *has a* special decryption *if the syntax of the* Dec *algorithm can be extended to work with a master secret key* msk *instead of a secret key* sk*, and for every security parameter* λ *and message* $x \in \{0,1\}^{n(\lambda)}$*:*

$$\Pr\left[\mathsf{Dec}_{\mathsf{msk}}\left(\mathsf{Enc}_{\mathsf{msk}}\left(x\right)\right) = x \,\middle|\, \mathsf{msk} \leftarrow \mathsf{Setup}\left(1^\lambda\right)\right] = 1$$

Similarly, we say that a public-key functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *has a* special decryption *if the syntax of the* Dec *algorithm can be extended to work with a master secret key* msk *instead of a secret key* sk*, and for every security parameter* λ *and message* $x \in \{0,1\}^{n(\lambda)}$*:*

$$\Pr\left[\mathsf{Dec}_{\mathsf{msk}}\left(\mathsf{Enc}_{\mathsf{mpk}}\left(x\right)\right) = x \,\middle|\, (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}\left(1^\lambda\right)\right] = 1$$

Constructions. We now summarize what is known about FE schemes as defined above. The following theorem essentially follows from prior work [SS10, GVW12] using the machinery of garbled circuits. For completeness, we describe the constructions in Appendix B.1 and B.2.

Theorem 2.5. *Under the assumption that standard public-key encryption schemes exist, there exists a single-key public-key functional-encryption scheme with the special decryption property for any polynomial-time function* F*. Under the assumption that one-way functions exist, there exists a single-key symmetric-key functional-encryption scheme with the special encryption and special decryption properties for any polynomial-time function* F*.*

There is some fixed polynomial $\mathsf{poly}(\lambda)$ *such that for a function* $F : \{0,1\}^{n_1(\lambda)} \times \{0,1\}^{n_2(\lambda)} \to \{0,1\}^{n_3(\lambda)}$ *with circuit size* $s(\lambda)$*, the resulting FE schemes have a master public key* mpk *(in the case of public-key FE), master secret key* msk*, and secret keys* sk_y *of size* $n_2(\lambda)\mathsf{poly}(\lambda)$ *and encryption/decryption time and ciphertext size* $s(\lambda)\mathsf{poly}(\lambda)$*. Assuming identity-based encryption (IBE) we can further reduce the size of* mpk *to be just* $\mathsf{poly}(\lambda)$*.*

See Appendix B for a proof of the above.

3 Invisible-Edits

We begin by defining and constructing encryption schemes with invisible edits. We start with the public key setting and then move on to the symmetric-key setting.

3.1 Public-Key Invisible-Edits

Our definition of public-key encryption with invisible edits follows the *dual-key* paradigm. The key generation algorithm outputs a "privileged" key pair $(\mathsf{pk}, \mathsf{sk})$ along with an edit key ek. The edit key can be used to generate an "unprivileged" key pair $(\mathsf{pk}_e, \mathsf{sk}_e) \leftarrow \mathsf{InvEdit}_{\mathsf{ek}}(e)$ corresponding to some edit e. An encryption of a message m encrypted under pk will decrypt to $m' = \mathsf{Edit}(m, e)$ under sk_e. A user cannot tell the difference between the following two scenarios:

- The user is an unprivilidged user that gets $(\mathsf{pk}_e, \mathsf{sk}_e)$ and sees encryptions $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_i)$ of messages m_i under the privileged public key pk which he decrypts incorrectly to $m'_i = \mathsf{Edit}(m_i, e)$ under sk_e.
- The user is a privilidged user that gets $(\mathsf{pk}, \mathsf{sk})$ and sees encryptions $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m'_i)$ of messages $m'_i = \mathsf{Edit}(m_i, e)$ under the privileged public key pk which he decrypts correctly to m'_i under sk.

The above even holds under chosen message attack where the user can choose the messages m_i. Note that since $(\mathsf{pk}, \mathsf{sk})$ and $(\mathsf{pk}_e, \mathsf{sk}_e)$ are indistinguishable it implies that correctness must hold when using the latter key pair and for any m with all but negligible probability $\mathsf{Dec}_{\mathsf{sk}_e}(\mathsf{Enc}_{\mathsf{pk}_e}(m)) = m$ since otherwise it would be easy to distinguish $(\mathsf{pk}_e, \mathsf{sk}_e)$ from $(\mathsf{pk}, \mathsf{sk})$.

Definition 3.1 (Public-Key Invisible Edits)**.** *An* Edit*-invisible public-key encryption with message-length* $n = n(\lambda)$*, edit description length* $\ell = \ell(\lambda)$*, and edit function* $\mathsf{Edit} : \{0,1\}^{n(\lambda)} \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{n(\lambda)}$ *consists of PPT algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{InvEdit})$ *with the following syntax:*

- $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda)$ *generates a public-key* pk*, secret-key* sk*, and edit key* ek*.*
- $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m), m = \mathsf{Dec}_{\mathsf{sk}}(c)$ *have the standard syntax of public-key encryption and decryption.*
- $(\mathsf{pk}_e, \mathsf{sk}_e) \leftarrow \mathsf{InvEdit}_{\mathsf{ek}}(e)$ *takes as input an edit e and outputs a public/secret key pair* $\mathsf{pk}_e, \mathsf{sk}_e$*.*

The scheme should satisfy the following properties:

Correctness & Encryption Security. *The scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *satisfies the standard notions of public-key encryption correctness and semantic security (see Definition A.1) if we ignore the edit-key* ek*.*

Invisibility of Edits. *We define the "invisible edits game"* $\mathsf{InvGame}^b_{\mathcal{A}}(\lambda)$ *between an adversary* $\mathcal{A}$ *and a challenger with a challenge bit* $b \in \{0,1\}$ *as follows:*

- *The adversary* $\mathcal{A}$ *chooses an edit function* $e \in \{0,1\}^\ell$*.*
- *Sample* $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda)$ *and* $(\mathsf{pk}_e, \mathsf{sk}_e) \leftarrow \mathsf{InvEdit}_{\mathsf{ek}}(e)$*. If* $b = 0$*, give* $(\mathsf{pk}, \mathsf{sk})$ *to* $\mathcal{A}$ *and let* $\mathcal{O}(\cdot) := \mathsf{Enc}_{\mathsf{pk}}(\mathsf{Edit}(\cdot, e))$*, else if* $b = 1$ *give* $(\mathsf{pk}_e, \mathsf{sk}_e)$ *to* $\mathcal{A}$ *and let* $\mathcal{O}(\cdot) := \mathsf{Enc}_{\mathsf{pk}}(\cdot)$*.*
- $\mathcal{A}^{\mathcal{O}}$ *gets access to the oracle* $\mathcal{O}$ *and eventually outputs a bit* b' *which we define as the output of the game.*

We require that for all PPT adversary $\mathcal{A}$ *we have* $|\Pr[\mathsf{InvGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{InvGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$*.*

3.1.1 Construction

We now present our construction of public-key invisible encryption using public-key FE. The construction follows the outline presented in the introduction. Before we give the construction, we define the function F_{Edit} which will be used throughout the paper.

Definition 3.2. *For every polynomial-time edit function* $\mathsf{Edit} : \{0,1\}^{n(\lambda)} \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{n(\lambda)}$*, we define the function* $F_{\mathsf{Edit}} : \{0,1\}^{n(\lambda)+(\lambda+\ell(\lambda))} \times \{0,1\}^{\lambda+\ell(\lambda)} \to \{0,1\}^{n(\lambda)}$ *as follows:*

$$F_{\mathsf{Edit}}(x = (m,k), y) := \begin{cases} \mathsf{Edit}(m, e) & \text{if } \exists e \text{ s.t. } y \oplus k = (0^\lambda, e) \\ m & \text{otherwise} \end{cases}$$

where we parse $x = (m, k)$ *with* $m \in \{0,1\}^{n(\lambda)}$ *and* $k \in \{0,1\}^{\lambda+\ell(\lambda)}$*.*

Construction 3.3 (Public-Key Invisible Edits)**.** For any polynomial-time edit function $\mathsf{Edit} : \{0,1\}^{n(\lambda)} \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{n(\lambda)}$, we construct an Edit-*invisible public-key encryption* using a single-key public-key functional encryption $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ for the function F_{Edit} (Definition 3.2). The construction proceeds as follows.

- $\mathsf{IEdit.Gen}(1^\lambda)$:
  - $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$
  - Select uniform $(y, k) \leftarrow \{0,1\}^{\lambda+\ell} \times \{0,1\}^{\lambda+\ell}$
  - $\mathsf{sk}_y \leftarrow \mathsf{FE.Gen}_{\mathsf{msk}}(y)$
  - Output $(\mathsf{pk} := (\mathsf{mpk}, k), \mathsf{sk} := \mathsf{sk}_y, \mathsf{ek} := (\mathsf{mpk}, k, \mathsf{msk}))$
- $\mathsf{IEdit.Enc}_{\mathsf{pk}}(m)$:
  - Output $c \leftarrow \mathsf{FE.Enc}_{\mathsf{mpk}}((m, k))$
- $\mathsf{IEdit.Dec}_{\mathsf{sk}}(c)$:
  - Output $m = \mathsf{FE.Dec}_{\mathsf{sk}_y}(c)$
- $\mathsf{IEdit.InvEdit}_{\mathsf{ek}}(e)$:
  - Select uniform $k' \leftarrow \{0,1\}^{\lambda+\ell}$
  - $\mathsf{sk}_{y'} \leftarrow \mathsf{FE.Gen}_{\mathsf{msk}}(y')$ where $y' = k \oplus (0^\lambda, e)$
  - Output $(\mathsf{pk}_e := (\mathsf{mpk}, k'), \mathsf{sk}_e := \mathsf{sk}_{y'})$

Theorem 3.4. *The scheme* IEdit *given in the above construction 3.3 is a secure* Edit-invisible public-key encryption *if* FE *is a single-key public-key functional encryption for the function* F_{Edit}*. In particular, the construction only relies on the existence of standard public-key encryption.*

Proof. We now prove that the above Construction 3.3 satisfies the properties of Edit-invisible public-key encryption in Definition 3.1.

Correctness: For every security parameter λ, and message $m \in \{0,1\}^n$:

$$\begin{aligned}
&\Pr\left[m = \mathsf{IEdit.Dec}_{\mathsf{sk}}(\mathsf{IEdit.Enc}_{\mathsf{pk}}(m)) \;\middle|\; (\mathsf{pk}, \mathsf{sk}, \mathsf{dk}) \leftarrow \mathsf{IEdit.Gen}(1^\lambda)\right] \\
&\quad = \Pr\left[m = \mathsf{FE.Dec}_{\mathsf{sk}_y}(\mathsf{FE.Enc}_{\mathsf{pk}}((m,k))) \;\middle|\; \begin{array}{l}(k,y) \leftarrow \{0,1\}^{\lambda+\ell} \times \{0,1\}^{\lambda+\ell} \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda) \\ \mathsf{sk}_y \leftarrow \mathsf{FE.Gen}_{\mathsf{msk}}(y)\end{array}\right] \\
&\quad = \Pr\left[m = F((m,k), y) \;\middle|\; (k,y) \leftarrow \{0,1\}^{\lambda+\ell} \times \{0,1\}^{\lambda+\ell}\right] \\
&\quad = 1 - \Pr\left[y \oplus k = (0^\lambda, r) \;\middle|\; \begin{array}{l}(k,y) \leftarrow \{0,1\}^{\lambda+\ell} \times \{0,1\}^{\lambda+\ell} \\ r \in \{0,1\}^\ell\end{array}\right] \\
&\quad = 1 - \frac{1}{2^\lambda}
\end{aligned}$$

Encryption Security: We want to show that for any PPT adversary $\mathcal{A}$:

$$|\Pr[\mathsf{CPAGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{CPAGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

This follows since an adversary $\mathcal{A}$ who breaks the CPA security also wins in the single-key public-key functional-encryption security game FEGame (with no secret key, when $y = \perp$).

Invisibility of Edits. We want to show that for any PPT adversary $\mathcal{A}$:

$$|\Pr[\mathsf{InvGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{InvGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

Informally, an adversary $\mathcal{A}$ who wins the "invisible edits game" $\mathsf{InvGame}^b_{\mathcal{A}}(\lambda)$ with an edit e and oracle queries m_i, wins the single-key public-key functional-encryption security game with a random $y \leftarrow \{0,1\}^{\lambda+\ell}$, and messages $x_0 = (\mathsf{Edit}(m_i, e), k)$ and $x_1 = (m_i, k')$ where $k \leftarrow \{0,1\}^{\lambda+\ell}$ and $k' = y \oplus (0^\lambda, e)$.

Formally, we prove it by a sequence of q hybrids where q is a bound of the number of queries that $\mathcal{A}$ makes to its oracle $\mathcal{O}$. We define the hybrid games $\mathsf{HybGame}^j_{\mathcal{A}}(\lambda)$ for $j = 0, \ldots, q$ by modifying $\mathsf{InvGame}^b_{\mathcal{A}}(\lambda)$ and defining the encryption oracle $\mathcal{O}^j$ and the challenge key pair $(\mathsf{pk}, \mathsf{sk})$ given to the adversary as follows:

$$\mathcal{O}^j(\cdot) := \begin{cases} \mathsf{FE.Enc}_{\mathsf{mpk}}(\mathsf{Edit}(\cdot, e), k) & i > j \\ \mathsf{FE.Enc}_{\mathsf{mpk}}(\cdot, k') & i \leq j \end{cases}$$

$$(\mathsf{pk}, \mathsf{sk}) := ((\mathsf{mpk}, k), \mathsf{sk}_y)$$

where $y, k \leftarrow \{0,1\}^{\lambda+\ell}$, $k' = y \oplus (0^\ell, e)$, and i is the index of the current query.

Observe that $\mathsf{HybGame}^0_{\mathcal{A}}(\lambda) \equiv \mathsf{InvGame}^0_{\mathcal{A}}(\lambda)$. This is because the value k used by the encryption oracle $\mathcal{O}$ matches the one in pk, the value y is random and independent of k, and the encryption oracle is encrypting edited messages.

Also observe that $\mathsf{HybGame}^q_{\mathcal{A}}(\lambda) \equiv \mathsf{InvGame}^1_{\mathcal{A}}(\lambda)$. This is because the value k' used by the encryption oracle is independent of the value k given in pk, the value y contained in sk is correlated to the value k' used by the encryption oracle via the relationship $y \oplus k' = (0^\ell, e)$, and the encryption oracle is encrypting un-edited messages.

Therefore, is suffices to show that for each j, the hybrids $\mathsf{HybGame}^j_{\mathcal{A}}(\lambda)$ and $\mathsf{HybGame}^{j+1}_{\mathcal{A}}(\lambda)$ are indistinguishable. This follows directly by public-key functional-encryption security. In particular, the only difference between the games is whether query $(j+1)$ to $\mathcal{O}$ is answered as $\mathsf{FE.Enc}_{\mathsf{mpk}}(\mathsf{Edit}(\cdot, e), k)$ or $\mathsf{FE.Enc}_{\mathsf{mpk}}(\cdot, k')$. But, since for any m we have $F(x_0, y) = F(x_1, y)$ where $x_0 = (\mathsf{Edit}(m, e), k)$, $x_1 = (m, k')$, this is indistinguishable by functional-encryption security.

□

3.2 Symmetric-Key Invisible-Edits

In the symmetric-key setting, we present two different definitions of encryption with invisible edits.

First, we present a definition that follows the *dual-key* paradigm and can be seen as a direct analogue of our public-key definition for the symmetric-key setting. We can always interpret a public-key encryption with invisible edits as a symmetric-key scheme and therefore we can achieve this definition assuming the existence of standard public-key encryption using the results from the previous section. However, it remains as a fascinating open problem whether one can construct symmetric-key encryption with invisible edits following the dual-key paradigm by relying only one one-way functions or whether public-key encryption is necessary.

Definition 3.5 (Dual-Key Invisible Edits)**.** *A* dual-key Edit-invisible symmetric-key encryption scheme *with message-length $n = n(\lambda)$, edit description length $\ell = \ell(\lambda)$, and edit function* $\mathsf{Edit} : \{0,1\}^n \times \{0,1\}^\ell \rightarrow \{0,1\}^n$ *consists of PPT algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{InvEdit})$ *with the following syntax:*

- $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda)$ *generates a secret-key* sk *and edit key* ek*.*
- $c \leftarrow \mathsf{Enc}_{\mathsf{sk}}(m), m = \mathsf{Dec}_{\mathsf{sk}}(c)$ *have the standard syntax of symmetric-key encryption and decryption.*
- $\mathsf{sk}_e \leftarrow \mathsf{InvEdit}_{\mathsf{ek}}(e)$ *takes as input an edit e and outputs a secret key* sk_e*.*

The scheme should satisfy the following properties:

Correctness & Encryption Security. *The scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *satisfies the standard notions of symmetric-key encryption correctness and CPA security (see Definition A.2) if we ignore the edit-key* ek*.*

Invisibility of Edits. *We define the "invisible edits game"* $\mathsf{InvGame}^b_{\mathcal{A}}(\lambda)$ *between an adversary* $\mathcal{A}$ *and a challenger with a challenge bit* $b \in \{0,1\}$ *as follows:*

- *The adversary* $\mathcal{A}$ *chooses an edit function* $e \in \{0,1\}^\ell$.
- *Sample* $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda)$ *and* $\mathsf{sk}_e \leftarrow \mathsf{InvEdit}_{\mathsf{ek}}(e)$. *If* $b = 0$, *let* $\mathcal{O}(\cdot) := \mathsf{Enc}_{\mathsf{sk}}(\mathsf{Edit}(\cdot, e))$ *and if* $b = 1$ *let* $\mathcal{O}(\cdot) := \mathsf{Enc}_{\mathsf{sk}}(\cdot)$.
- $\mathcal{A}^{\mathcal{O}}$ *gets the secret key* sk *if* $b = 0$, *and* sk_e *if* $b = 1$ *together with an access to the oracle* $\mathcal{O}$. *Eventually* $\mathcal{A}$ *outputs a bit* b' *which we define as the output of the game.*

We require that for all PPT adversary $\mathcal{A}$ *we have* $|\Pr[\mathsf{InvGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{InvGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.

Below, we present a definition of symmetric-key encryption with invisible edits that follows the weaker *dual-scheme* paradigm. In this case there are two different encryption schemes: an *unprivileged* scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a *privileged* scheme $(\mathsf{PrivGen}, \mathsf{PrivEnc}, \mathsf{PrivDec})$. Given a secret key sk^* for the privileged scheme, it's possible to create a secret key $\mathsf{sk}_e \leftarrow \mathsf{InvEdit}_{\mathsf{sk}^*}(e)$ that looks like a legitimate secret key of the unprivileged scheme but is tied to some edit e. An encryption of a message m encrypted under sk^* will decrypt to $m' = \mathsf{Edit}(m, e)$ under sk_e. A user cannot tell the difference between the following two scenarios:

- The user is an unprivileged user that gets $\mathsf{sk}_e \leftarrow \mathsf{InvEdit}_{\mathsf{sk}^*}(e)$ and sees encryptions $c_i \leftarrow \mathsf{Enc}_{\mathsf{sk}^*}(m_i)$ of messages m_i under the privileged secret key sk^* which he decrypts incorrectly to $m'_i = \mathsf{Edit}(m_i, e)$ under sk_e.
- The user is an unprivilidged user that gets $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$ created using the legitimate unprivileged key generation algorithm and sees encryptions $c_i \leftarrow \mathsf{Enc}_{\mathsf{sk}}(m'_i)$ of messages $m'_i = \mathsf{Edit}(m_i, e)$ under the unprivilidged secret key sk which he then decrypts correctly to m'_i using the same sk.

In other words, the user can tell that he's unprivileged. But he does't know whether everyone else is also unprivileged and he's correctly decrypting the messages they are sending or whether some other users are privileged and he's decrypting edited messages.

Definition 3.6 (Dual-Scheme Invisible Edits)**.** *A dual-scheme* Edit*-invisible symmetric-key encryption scheme with message-length* $n = n(\lambda)$, *edit description length* $\ell = \ell(\lambda)$, *and edit function* $\mathsf{Edit} : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^n$ *consists of PPT algorithms* $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{PrivGen}, \mathsf{PrivEnc}, \mathsf{PrivDec}, \mathsf{InvEdit}$. *The schemes* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, $(\mathsf{PrivGen}, \mathsf{PrivEnc}, \mathsf{PrivDec})$ *have the usual symmetric-key encryption syntax. The algorithm* $\mathsf{sk}_e \leftarrow \mathsf{InvEdit}_{\mathsf{sk}}(e)$ *takes as input an edit* e *and a privileged secret key* sk *and outputs an unprivileged secret key* sk_e *tied to an edit* e.

Correctness & Encryption Security. *The schemes* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *and* $(\mathsf{PrivGen}, \mathsf{PrivEnc}, \mathsf{PrivDec})$ *satisfy the standard notions of symmetric-key encryption correctness and CPA security (Definition A.2).*

Invisibility of Edits. *We define the "invisible edits game"* $\mathsf{InvGame}^b_{\mathcal{A}}(\lambda)$ *between an adversary* $\mathcal{A}$ *and a challenger with a challenge bit* $b \in \{0,1\}$ *as follows:*

- *The adversary* $\mathcal{A}$ *chooses an edit function* $e \in \{0,1\}^\ell$.
- *If* $b = 0$ *sample* $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$ *and if* $b = 1$ *sample* $\mathsf{sk}^* \leftarrow \mathsf{PrivGen}(1^\lambda)$ *and* $\mathsf{sk}_e \leftarrow \mathsf{InvEdit}_{\mathsf{sk}^*}(e)$. *If* $b = 0$, *let* $\mathcal{O}(\cdot) := \mathsf{Enc}_{\mathsf{sk}}(\mathsf{Edit}(\cdot, e))$ *and if* $b = 1$ *let* $\mathcal{O}(\cdot) := \mathsf{PrivEnc}_{\mathsf{sk}^*}(\cdot)$.
- *The adversary* $\mathcal{A}^{\mathcal{O}}$ *gets the secret key* sk *if* $b = 0$, *or* sk_e *if* $b = 1$. *It also gets oracle access to* $\mathcal{O}(\cdot)$ *and eventually it outputs a bit* b' *which we define as the output of the game.*

We require that for all PPT adversary $\mathcal{A}$ *we have* $|\Pr[\mathsf{InvGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{InvGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.

3.2.1 Construction

Our construction for the dual-scheme symmetric-key encryption with invisible edits roughly follows the same outline as the public-key construction with the main difference that we rely on symmetric-key rather than public-key FE.

Construction 3.7 (Dual-Scheme Invisible Edits)**.** For any polynomial time edit function $\mathsf{Edit} : \{0,1\}^{n(\lambda)} \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{n(\lambda)}$, we construct a *dual-scheme* Edit*-invisible symmetric-key encryption* $\mathsf{DSInvE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{PrivGen}, \mathsf{PrivEnc}, \mathsf{PrivDec}, \mathsf{InvEdit})$, using a single-key symmetric-key functional encryption $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with special encryption (see Definition 2.3) for the function $F := F_{\mathsf{Edit}}$ (see Definition 3.2). The construction proceeds as follows.

$\mathsf{DSInvE.PrivGen}(1^\lambda)$:

- $\mathsf{FE.msk} \leftarrow \mathsf{FE.Setup}(1^\lambda)$
- Select uniform $k^* \leftarrow \{0,1\}^{\lambda+\ell}$
- Output $\mathsf{sk}^* = (\mathsf{FE.msk}, k^*)$

$\mathsf{DSInvE.PrivEnc}_{\mathsf{sk}^*}(m)$:

- Output $c \leftarrow \mathsf{FE.Enc}_{\mathsf{msk}}((m, k^*))$

$\mathsf{DSInvE.PrivDec}_{\mathsf{sk}^*}(c)$:

- $(m, k^*) = \mathsf{FE.Dec}_{\mathsf{msk}}(c)$
- Output m

$\mathsf{DSInvE.InvEdit}_{\mathsf{sk}^*}(e)$:

- $\mathsf{FE.sk}_{y'} \leftarrow \mathsf{FE.Gen}_{\mathsf{msk}}(y')$ where $y' = k^* \oplus (0^\lambda, e)$
- Output $\mathsf{sk}_e = \mathsf{FE.sk}_{y'}$

$\mathsf{DSInvE.Gen}(1^\lambda)$:

- $\mathsf{FE.msk} \leftarrow \mathsf{FE.Setup}(1^\lambda)$
- Select uniform $y \leftarrow \{0,1\}^{\lambda+\ell}$
- $\mathsf{FE.sk}_y \leftarrow \mathsf{FE.Gen}_{\mathsf{msk}}(y)$
- Output $\mathsf{sk} = \mathsf{FE.sk}_y$

$\mathsf{DSInvE.Enc}_{\mathsf{sk}}(m)$:

- Select uniform $k \leftarrow \{0,1\}^{\lambda+\ell}$
- Output $c \leftarrow \mathsf{FE.Enc}_{\mathsf{sk}_y}((m, k))$

$\mathsf{DSInvE.Dec}_{\mathsf{sk}}(c)$:

- Output $m = \mathsf{FE.Dec}_{\mathsf{sk}_y}(c)$

Theorem 3.8. *The scheme* DSInvE *given in the above construction 3.7 is a secure* dual-scheme Edit-invisible symmetric-key encryption *if* FE *is a single-key symmetric-key functional encryption scheme with special encryption for the function* F_{Edit}*. In particular, the construction only relies on the existence of one-way functions.*

Proof. We now prove that Construction 3.7 satisfies the properties of *dual-scheme* Edit*-invisible symmetric-key encryption* in Definition 3.6.

Correctness For every security parameter λ, and message $m \in \{0,1\}^n$:

$$\begin{aligned}
&\Pr\left[m = \mathsf{DSInvE.Dec}_{\mathsf{sk}}(\mathsf{DSInvE.Enc}_{\mathsf{sk}}(m)) \;\middle|\; \mathsf{sk} \leftarrow \mathsf{DSInvE.Gen}(1^\lambda)\right] \\
&= \Pr\left[m = \mathsf{FE.Dec}_{\mathsf{sk}_y}(\mathsf{FE.Enc}_{\mathsf{sk}_y}((m,k))) \;\middle|\; \begin{array}{l}(k,y) \leftarrow \{0,1\}^{\lambda+\ell} \times \{0,1\}^{\lambda+\ell} \\ \mathsf{msk} \leftarrow \mathsf{FE.Setup}(1^\lambda) \\ \mathsf{sk}_y \leftarrow \mathsf{FE.Gen}_{\mathsf{msk}}(y)\end{array}\right] \\
&= \Pr\left[m = F((m,k),y) \;\middle|\; (k,y) \leftarrow \{0,1\}^{\lambda+\ell} \times \{0,1\}^{\lambda+\ell}\right] \\
&= 1 - \Pr\left[\exists e \in \{0,1\}^\ell : y \oplus k = (0^\lambda, e) \;\middle|\; (k,y) \leftarrow \{0,1\}^{\lambda+\ell} \times \{0,1\}^{\lambda+\ell}\right] \\
&= 1 - \frac{1}{2^\lambda}
\end{aligned}$$

Therefore, the scheme $(\mathsf{DSInvE.Gen}, \mathsf{DSInvE.Enc}, \mathsf{DSInvE.Dec})$ is correct. Moreover, for every security parameter λ, and message $m \in \{0,1\}^n$:

$$\begin{aligned}
&\Pr\left[m = \mathsf{DSInvE.PrivDec}_{\mathsf{sk}^*}(\mathsf{DSInvE.PrivEnc}^*_{\mathsf{sk}}(m)) \;\middle|\; \mathsf{sk}^* \leftarrow \mathsf{DSInvE.PrivGen}(1^\lambda)\right] \\
&= \Pr\left[m = \mathsf{FE.Dec}_{\mathsf{msk}}(\mathsf{FE.Enc}_{\mathsf{msk}}((m,k^*))) \;\middle|\; \begin{array}{l}k^* \leftarrow \{0,1\}^{\lambda+\ell} \\ \mathsf{msk} \leftarrow \mathsf{FE.Setup}(1^\lambda)\end{array}\right] \\
&= 1
\end{aligned}$$

Thus, also the scheme$(\mathsf{DSInvE.PrivGen}, \mathsf{DSInvE.PrivEnc}, \mathsf{DSInvE.PrivDec})$ is correct.

Encryption Security. The scheme $(\mathsf{DSInvE.PrivGen}, \mathsf{DSInvE.PrivEnc}, \mathsf{DSInvE.PrivDec})$ is symmetrically secure (i.e., CPA secure). Namely, for every PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every security parameter λ,

$$|\Pr[\mathsf{CPAGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{CPAGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

A PPT adversary $\mathcal{A}$ who wins the CPA security also wins the single-key symmetric-key functional-encryption security game FEGame (with no secret key).

The scheme $(\mathsf{DSInvE.Gen}, \mathsf{DSInvE.Enc}, \mathsf{DSInvE.Dec})$ is also symmetrically secure. The underline functional encryption scheme FE has a special encryption, therefore no PPT can distinguish between the CPA game $\mathsf{CPAGame}^b_{\mathcal{A}}(\lambda)$ and the hybrid game $\mathsf{HybGame}^b_{\mathcal{A}}(\lambda)$ where the encryption oracle and the challenge ciphertext instead of:

$$\mathcal{O}(\cdot) := \mathsf{FE.Enc}_{\mathsf{sk}_y}((\cdot,k))_{k \leftarrow \{0,1\}^{\lambda+\ell}} \qquad c \leftarrow \mathsf{DSInvE.Enc}_{\mathsf{sk}}(\cdot) := \mathsf{FE.Enc}_{\mathsf{sk}_y}((m_b,k))_{k \leftarrow \{0,1\}^{\lambda+\ell}}$$

are replaced with:

$$\mathcal{O}(\cdot) := \mathsf{FE.Enc}_{\mathsf{msk}}((\cdot,k))_{k \leftarrow \{0,1\}^{\lambda+\ell}} \qquad c \leftarrow \mathsf{DSInvE.Enc}_{\mathsf{sk}}(\cdot) := \mathsf{FE.Enc}_{\mathsf{msk}}((m_b,k))_{k \leftarrow \{0,1\}^{\lambda+\ell}}$$

The $\mathsf{HybGame}$ game is the same as the FEGame (with no secret-key, when $y = \perp$).

Invisibility of Edits. For any PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$, such that for every security parameter λ

$$|\Pr[\mathsf{InvGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{InvGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

An PPT adversary who wins the "invisible edits game" $\mathsf{InvGame}^b_{\mathcal{A}}(\lambda)$ with an edit e and an oracle query m_i, wins the single-key symmetric-key functional encryption security with a random $y \leftarrow \{0,1\}^{\lambda+\ell}$ and messages $m_0 = (\mathsf{Edit}(m_{\tilde{q}}, e), k)$ and $m_1 = (m_{\tilde{q}}, k^*)$ where $k \leftarrow \{0,1\}^{\lambda+\ell}$ and $k^* = y \oplus (0^\ell, e)$.

Formally, we prove it by a sequence of $(q+1)$ hybrids where q is a bound of the number of query messages that $\mathcal{A}$ is able to make.

We define the hybrid game $\mathsf{HybGame}^{\tilde{q},b}_{\mathcal{A}}(\lambda)$ (a modification of $\mathsf{InvGame}^{b}_{\mathcal{A}}$), in which the encryption oracle $\mathcal{O}(\cdot)$ and challenge sk are:

$$\mathcal{O}^{\tilde{q}}_{b}(\cdot) := \begin{cases} \mathsf{FE.Enc}_{\mathsf{msk}}(\mathsf{Edit}(\cdot, e), k)_{k \leftarrow \{0,1\}^{\lambda+\ell}} & i < \tilde{q} \\ \mathsf{FE.Enc}_{\mathsf{msk}}(\mathsf{Edit}(m_{\tilde{q}}, e), k)_{k \leftarrow \{0,1\}^{\lambda+\ell}} & b = 0 \wedge i = \tilde{q} \\ \mathsf{FE.Enc}_{\mathsf{msk}}(m_{\tilde{q}}, k^*) & b = 1 \wedge i = \tilde{q} \\ \mathsf{FE.Enc}_{\mathsf{msk}}(\cdot, k^*) & i > \tilde{q} \end{cases} \qquad \mathsf{sk} \leftarrow \mathsf{FE.Gen}_{\mathsf{msk}}(y)$$

where $y \leftarrow \{0,1\}^{\lambda+\ell}$, $k^* = y \oplus (0^\ell, e)$, and i is the number of queries that were asked.

By the public-key functional-encryption security[2] it holds that for every $\tilde{q} \in [q]$ and every PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$, such that for every security parameter λ

$$|\Pr[\mathsf{HybGame}^{\tilde{q},0}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{HybGame}^{\tilde{q},1}_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda). \tag{1}$$

Note that syntactically for every $\tilde{q}$:

$$\mathsf{HybGame}^{\tilde{q},1}_{\mathcal{A}}(\lambda) = \mathsf{HybGame}^{(\tilde{q}-1),0}_{\mathcal{A}}(\lambda). \tag{2}$$

Hybrid 0: we start with the invisibility game with $b = 0$, $\mathsf{InvGame}^{0}_{\mathcal{A}}(\lambda)$. The encryption oracle and challenge are:

$$\begin{aligned} \mathcal{O}(\cdot) &:= \mathsf{IEdit.Enc}_{\mathsf{sk}}(\mathsf{Edit}(\cdot, e)) && = \mathsf{FE.Enc}_{\mathsf{sk}_y}(\mathsf{Edit}(\cdot, e), k)_{k \leftarrow \{0,1\}^{\lambda+\ell}} \\ \mathsf{sk} &\leftarrow \mathsf{IEdit.Gen}(1^\lambda) && = \mathsf{sk}_y \end{aligned}$$

where $y \leftarrow \{0,1\}^{\lambda+\ell}$, $\mathsf{sk}_y \leftarrow \mathsf{FE.Gen}_{\mathsf{msk}}(y)$, and $\mathsf{msk} \leftarrow \mathsf{FE.Setup}(1^\lambda)$.

Hybrid 1: we move to a the hybrid game $\mathsf{HybGame}^{q,0}_{\mathcal{A}}(\lambda)$ in which we encrypt using $\mathsf{FE.Enc}_{\mathsf{msk}}$ (instead of using $\mathsf{FE.Enc}_{\mathsf{sk}_y}$). The encryption oracle and the challenge are:

$$\begin{aligned} \mathcal{O}(\cdot) &:= \mathsf{FE.Enc}_{\mathsf{msk}}(\mathsf{Edit}(\cdot, e), k)_{k \leftarrow \{0,1\}^{\lambda+\ell}} \\ \mathsf{sk} &:= \mathsf{FE.Gen}_{\mathsf{msk}}(y) \end{aligned}$$

where $y \leftarrow \{0,1\}^{\lambda+\ell}$, and $\mathsf{msk} \leftarrow \mathsf{FE.Setup}(1^\lambda)$. By the special encryption property[3],

$$\mathsf{InvGame}^{0}_{\mathcal{A}}(\lambda) \overset{c}{\approx} \mathsf{HybGame}^{q,0}_{\mathcal{A}}(\lambda)$$

.

Hybrid 2: we move to the hybrid game $\mathsf{HybGame}^{1,1}_{\mathcal{A}}(\lambda)$ by a sequence of q hybrids (each start with equation (1) and follows by equation (2)). Namely,

$$\begin{gathered} \mathsf{HybGame}^{q,0}_{\mathcal{A}}(\lambda) \overset{c}{\approx} \mathsf{HybGame}^{q,1}_{\mathcal{A}}(\lambda) = \mathsf{HybGame}^{(q-1),0}_{\mathcal{A}}(\lambda) \\ \mathsf{HybGame}^{(q-1),0}_{\mathcal{A}}(\lambda) \overset{c}{\approx} \mathsf{HybGame}^{(q-1),1}_{\mathcal{A}}(\lambda) = \mathsf{HybGame}^{(q-2),0}_{\mathcal{A}}(\lambda) \\ \vdots \\ \mathsf{HybGame}^{1,0}_{\mathcal{A}}(\lambda) \overset{c}{\approx} \mathsf{HybGame}^{1,1}_{\mathcal{A}}(\lambda) \end{gathered}$$

Observe that $\mathsf{HybGame}^{1,1}_{\mathcal{A}}(\lambda) = \mathsf{InvGame}^{1}_{\mathcal{A}}(\lambda)$ in which the encryption oracle $\mathcal{O}$ and challenge sk are:

$$\begin{aligned} \mathcal{O}(\cdot) &:= \mathsf{IEdit.PrivEnc}_{\mathsf{sk}^*}(\cdot) && = \mathsf{FE.Enc}_{\mathsf{msk}}(\cdot, k^*) \\ \mathsf{sk}_e &\leftarrow \mathsf{IEdit.InvEdit}_{\mathsf{sk}^*}(e) && = \mathsf{FE.Gen}_{\mathsf{msk}}(y) \end{aligned}$$

where $y = k^* \oplus (0^\ell, e)$, $k^* \leftarrow \{0,1\}^{\lambda+\ell}$, and $\mathsf{msk} \leftarrow \mathsf{FE.Setup}(1^\lambda)$.

□

[2] with random $y \leftarrow \{0,1\}^{\lambda+\ell}$ and $m_0 = (\mathsf{Edit}(m_{\tilde{q}}, e), k), m_1 = (m_{\tilde{q}}, k^*)$ where $y, k \leftarrow \{0,1\}^{\lambda+\ell}$ and $k^* = y \oplus (0^\ell, e)$

[3] see Definition 2.3.

3.3 Efficiency

For an edit function $\mathsf{Edit} : \{0,1\}^{n(\lambda)} \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{n(\lambda)}$ with $n(\lambda)$ size message, $\ell(\lambda)$ edit description size, and where the circuit size of Edit is $s(\lambda)$ the efficiency of our public-key and symmetric-key Edit-invisible encryption schemes given in Constructions 3.3 and 3.7 is summarized as follows. There is some fixed polynomial $\mathsf{poly}(\lambda)$ such that:

- The secret key size is $\ell(\lambda)\mathsf{poly}(\lambda)$.
- The run-time of the encryption/decryption procedures and the ciphertext size $s(\lambda)\mathsf{poly}(\lambda)$.
- In the case of public-key deniable encryption, the public-keys size is $\ell(\lambda)\mathsf{poly}(\lambda)$. If we're willing to use a stronger assumption of *identity-based encryption* (IBE) we can use a more efficient FE instantiation in our construction which reduces the public-key size to just $\mathsf{poly}(\lambda)$.

One open problem would be to reduce the encryption time and the ciphertext size to only depend on the message size $n(\lambda)$ rather than the circuit size $s(\lambda)$ without increasing the secret key size. However, we envision that most interesting Edit functions that we'd want to apply anyway have relatively small circuit size which is roughly linear in the message size $s(\lambda) = O(n(\lambda))$.

4 Deniable-Edits

We now define and construct encryption schemes with deniable edits. We start with the public key setting and then move on to the symmetric-key setting.

4.1 Public-Key Deniable-Edits

Our definition of public-key encryption with deniable edits follows the *dual-key* paradigm outlined in the introduction. The key generation algorithm outputs a secret decryption key sk and denying key dk, and most users are expected to discard dk since it is not needed for decryption. In particular, this means that users might be later coerced to give up their secret key sk which they need to keep for decryption but we assume they cannot be coerced to give up dk since they can plausibly claim they discarded it. Users that keep dk can use it to later "deny" the contents of a particular ciphertext c encrypting some message m by producing a legitimate secret key $\mathsf{sk}_{c,e} \leftarrow \mathsf{Deny}_{\mathsf{dk}}(c,e)$ that decrypts c to $m' = \mathsf{Edit}(m,e)$. Given a ciphertext c and a secret key sk^*, the coercer cannot distinguish whether c is really an encryption of m' and $\mathsf{sk}^* = \mathsf{sk}$ is the original secret key output by the key generation algorithm or whether c is an encryption of m and $\mathsf{sk}^* = \mathsf{sk}_{c,e}$ is the modified secret key output by the denying algorithm.

Definition 4.1 (Public-Key Deniable Edits)**.** *An* Edit*-deniable public-key encryption with message-length* $n(\lambda)$*, edit description length* $\ell(\lambda)$*, and a PPT edit function* $\mathsf{Edit} : \{0,1\}^{n(\lambda)} \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{n(\lambda)}$ *consists of PPT algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Deny})$ *having the following syntax:*

- $(\mathsf{pk}, \mathsf{sk}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *generates a public-key* pk*, secret-key* sk *and denying key* dk*.*
- $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m), m = \mathsf{Dec}_{\mathsf{sk}}(c)$ *have the standard syntax of public-key encryption and decryption.*
- $\mathsf{sk}_{c,e} \leftarrow \mathsf{Deny}_{\mathsf{dk}}(c,e)$ *takes as input some ciphertext c encrypting data m along with an edit e and outputs a secret key* $\mathsf{sk}_{c,e}$ *that decrypts c to* $m' = \mathsf{Edit}(m,e)$*.*

The scheme should satisfy the following properties:

Encryption Correctness & Security: *The scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *should satisfy the standard correctness and CPA security definitions of public-key encryption (see Definition A.1), if we ignore the denying key* dk*.*

Deniability Security. *We define the "deniability game"* $\mathsf{DenGame}^b_{\mathcal{A}}(\lambda)$ *between an adversary* $\mathcal{A}$ *and a challenger with a challenge bit* $b \in \{0,1\}$ *as follows:*

- *Sample* $(\mathsf{pk}, \mathsf{sk}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *and give* pk *to* $\mathcal{A}$*.*

- *$\mathcal{A}$ chooses a message $m \in \{0,1\}^{n(\lambda)}$ and an edit $e \in \{0,1\}^{\ell(\lambda)}$.*
- *If $b = 0$, sample $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mathsf{Edit}(m,e))$ and give (sk, c) to $\mathcal{A}$.*
  If $b = 1$, sample $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$, $\mathsf{sk}_{c,e} \leftarrow \mathsf{Deny}_{\mathsf{dk}}(c,e)$ and give $(\mathsf{sk}_{c,e}, c)$ to $\mathcal{A}$.
- *$\mathcal{A}$ outputs a bit b' which we define as the output of the game.*

For all PPT adversary $\mathcal{A}$, we require $|\Pr[\mathsf{DenGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{DenGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.

4.1.1 Construction

The construction of public-key encryption with deniable edits is similar to our earlier construction of public-key encryption with invisible edits. The main difference is that we previously chose a random value k in the public key and use it for all encryptions whereas we now chose a fresh random value k during each encryption operation. The secret key sk_y is associated with a random value y. When we want to deny a particular ciphertext c which was created using k, we create a new secret key $\mathsf{sk}_{y'}$ with $y' = k \oplus (0^\lambda, e)$. This ensures that the edit is applied when decrypting the cihertext c via $\mathsf{sk}_{y'}$.

Construction 4.2 (Public-Key Deniable Edit)**.** For any polynomial-time edit function $\mathsf{Edit} : \{0,1\}^{n(\lambda)} \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{n(\lambda)}$, we construct an Edit-*deniable public-key encryption* $\mathsf{DEdit} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Deny})$ using a single-key public-key functional encryption $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with special decryption (see Definition 2.4) for the function $F := F_{\mathsf{Edit}}$ (see Definition 3.2). The construction proceeds as follows.

$\mathsf{DEdit.Gen}(1^\lambda)$:

- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$
- Select uniform $y \leftarrow \{0,1\}^{\lambda+\ell}$
- $\mathsf{sk}_y \leftarrow \mathsf{FE.Gen}_{\mathsf{msk}}(y)$
- Output $(\mathsf{pk} := \mathsf{mpk}, \mathsf{sk} := \mathsf{sk}_y, \mathsf{dk} := \mathsf{msk})$

$\mathsf{DEdit.Dec}_{\mathsf{sk}}(c)$:

- Output $m := \mathsf{FE.Dec}_{\mathsf{sk}_y}(c)$

$\mathsf{DEdit.Enc}_{\mathsf{pk}}(m)$:

- Select uniform $k \leftarrow \{0,1\}^{\lambda+\ell}$
- Output $c \leftarrow \mathsf{FE.Enc}_{\mathsf{mpk}}((m,k))$

$\mathsf{DEdit.Deny}_{\mathsf{dk}}(c,e)$:

- $(m,k) = \mathsf{FE.Dec}_{\mathsf{msk}}(c)$
- $\mathsf{sk}_y \leftarrow \mathsf{FE.Gen}_{\mathsf{msk}}(y)$ where $y = k \oplus (0^\lambda, e)$
- Output $\mathsf{sk}_{c,e} := \mathsf{sk}_y$

Theorem 4.3. *The scheme* DEdit *given in the above construction 4.2 is a secure* Edit*-deniable public-key encryption if* FE *is a single-key public-key functional encryption with special decryption for the function* F_{Edit}*. In particular, the construction only relies on the existence of standard public-key encryption.*

Proof. We now prove that Construction 4.2 satisfies the properties in Definition 4.1

Correctness: For every security parameter λ, and message $m \in \{0,1\}^n$:

$$\begin{aligned}
&\Pr\left[m = \mathsf{DEdit.Dec}_{\mathsf{sk}}(\mathsf{DEdit.Enc}_{\mathsf{pk}}(m)) \,\middle|\, (\mathsf{pk},\mathsf{sk},\mathsf{dk}) \leftarrow \mathsf{DEdit.Gen}(1^\lambda)\right] \\
&= \Pr\left[m = \mathsf{FE.Dec}_{\mathsf{sk}_y}(\mathsf{FE.Enc}_{\mathsf{mpk}}((m,k))) \,\middle|\, \begin{array}{l} (k,y) \leftarrow \{0,1\}^{\lambda+\ell} \times \{0,1\}^{\lambda+\ell} \\ (\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda) \\ \mathsf{sk}_y \leftarrow \mathsf{FE.Gen}_{\mathsf{msk}}(y) \end{array}\right] \\
&= \Pr\left[m = F((m,k),y) \,\middle|\, (k,y) \leftarrow \{0,1\}^{\lambda+\ell} \times \{0,1\}^{\lambda+\ell}\right] \\
&= 1 - \Pr\left[\exists e \in \{0,1\}^\ell : y \oplus k = (0^\lambda, e) \,\middle|\, (k,y) \leftarrow \{0,1\}^{\lambda+\ell} \times \{0,1\}^{\lambda+\ell}\right] \\
&= 1 - \frac{1}{2^\lambda}
\end{aligned}$$

Encryption Security: We want to show that for any PPT adversary $\mathcal{A}$:

$$|\Pr[\mathsf{CPAGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{CPAGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

This follows since an adversary $\mathcal{A}$ who breaks the CPA security also breaks the single-key public-key functional-encryption security game FEGame (with no secret key, when $y = \bot$).

Deniability Security. We want to show that for any PPT adversary $\mathcal{A}$:

$$|\Pr[\mathsf{DenGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{DenGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

This follows since an adversary $\mathcal{A}$ who wins the deniability game $\mathsf{DenGame}$ with message m and edit e, also wins the single-key public-key functional-encryption security game FEGame with random $y \leftarrow \{0,1\}^{\lambda+\ell}$, and messages $x_0 = (m, k)$ and $x_1 = (\mathsf{Edit}(m,e), k')$ where $k = y \oplus (0^{\lambda+\ell}, e)$, and $k' \leftarrow \{0,1\}^{\lambda+\ell}$. Note that $F_{\mathsf{Edit}}(x_0, y) = F_{\mathsf{Edit}}(x_1, y) = \mathsf{Edit}(m,e)$ unless $k' \oplus y = (0^\lambda, e')$ for some e' which happens with negligible probability. Formally, we construct $\mathcal{A}'$ (who uses an adversary $\mathcal{A}$ that wins in the $\mathsf{DenGame}$) to win the FEGame:

- The challenger samples $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}(\lambda)$ and sends mpk to $\mathcal{A}'$.
  The adversary $\mathcal{A}'$ chooses a random $y \leftarrow \{0,1\}^{\lambda+\ell}$.
- The challenger samples $\mathsf{sk}_y \leftarrow \mathsf{FE.Gen}_{\mathsf{msk}}(y)$ and sends sk_y to $\mathcal{A}'$.
  The adversary $\mathcal{A}'$ forward $\mathsf{pk} := \mathsf{mpk}$ to $\mathcal{A}$ and receives back a message $m \in \{0,1\}^n$ and an edit $e \in \{0,1\}^\ell$. The adversary $\mathcal{A}'$ chooses two messages $x_0 = (m, k)$ and $x_1 = (\mathsf{Edit}(m,e), k')$ where $k' \leftarrow \{0,1\}^{\lambda+\ell}$ and $k = y \oplus (0^{\lambda+\ell}, e)$.
- The challenge samples a ciphertext $c \leftarrow \mathsf{FE.Enc}_{\mathsf{mpk}}(m_b)$ and sends c to $\mathcal{A}'$.
  The adversary $\mathcal{A}'$ forwards (sk_y, c) to $\mathcal{A}$ and receives back a bit b' to output.

The advantage of $\mathcal{A}'$ in FEGame is the same as the advantage of $\mathcal{A}$ in $\mathsf{DenGame}$, up to the negligible probability that $k' \oplus y = (0^\lambda, e')$ for some e'.

□

4.2 Symmetric-Key Deniable-Edits

In the symmetric-key setting, we present two different definitions of encryption with deniable edits analogously to our two definitions of symmetric-key encryption with invisible edits.

First, we present a definition that follows the *dual-key* paradigm and can be seen as a direct analogue of our public-key definition for the symmetric-key setting. In particular, the key generation algorithm outputs a secret decryption key sk and denying key dk which can use it to later "deny" the contents of a particular ciphertext c encrypting some message m by producing a legitimate secret key $\mathsf{sk}_{c,e} \leftarrow \mathsf{Deny}_{\mathsf{dk}}(c,e)$ that decrypts c to $m' = \mathsf{Edit}(m,e)$. For both encryption security and deniability we assume that the adversary has access to an encryption oracle. We can always interpret a public-key deniable encryption scheme as a symmetric-key one and therefore we can achieve this definition assuming the existence of standard public-key encryption using the results from the previous section. However, it remains as a fascinating open problem whether one can construct symmetric-key deniable encryption following the dual-key paradigm by relying only one one-way functions or whether public-key encryption is necessary.

Definition 4.4 (Dual-Key Deniable Edits). *A dual-key* Edit*-deniable symmetric-key encryption scheme with message-length* $n(\lambda)$*, edit description length* $\ell(\lambda)$*, and edit function* $\mathsf{Edit} : \{0,1\}^{n(\lambda)} \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{n(\lambda)}$ *consists of PPT algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Deny})$ *with the following syntax:*

- $(\mathsf{sk}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *generates a secret-key* sk *and deniability key* dk.
- $c \leftarrow \mathsf{Enc}_{\mathsf{sk}}(m), m = \mathsf{Dec}_{\mathsf{sk}}(c)$ *have the standard syntax of public-key encryption and decryption.*
- $\mathsf{sk}_{c,e} \leftarrow \mathsf{Deny}_{\mathsf{dk}}(c,e)$ *takes as input some ciphertext* c *encrypting data* m *along with an edit* e *and outputs a secret key* $\mathsf{sk}_{c,e}$ *that decrypts* c *to* $m' = \mathsf{Edit}(m,e)$.

The scheme should satisfy the following properties:

Correctness & Encryption Security. *The scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *satisfies the standard notions of symmetric-key encryption correctness and CPA security (see Definition A.2) if we ignore the key* dk.

Deniability. *We define the "deniability game"* $\mathsf{DenGame}^b_{\mathcal{A}}(\lambda)$ *between an adversary* $\mathcal{A}$ *and a challenger with a challenge bit* $b \in \{0,1\}$ *as follows:*

- *Sample* $(\mathsf{sk}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
- $\mathcal{A}^{\mathsf{Enc}_{\mathsf{sk}}(\cdot)}$ *gets access to the encryption oracle. Eventually, it chooses a message* $m \in \{0,1\}^{n(\lambda)}$ *and an edit* $e \in \{0,1\}^{\ell(\lambda)}$.
- *If* $b = 0$*, sample* $c \leftarrow \mathsf{Enc}_{\mathsf{sk}}(\mathsf{Edit}(m, e))$ *and give* (sk, c) *to* $\mathcal{A}$.
  If $b = 1$*, sample* $c \leftarrow \mathsf{Enc}_{\mathsf{sk}}(m)$, $\mathsf{sk}_{c,e} \leftarrow \mathsf{Deny}_{\mathsf{dk}}(c, e)$ *and give* $(\mathsf{sk}_{c,e}, c)$ *to* $\mathcal{A}$.
- $\mathcal{A}^{\mathsf{Enc}_{\mathsf{sk}}(\cdot)}$ *outputs a bit* b' *which we define as the output of the game.*

We require that for all PPT adversary $\mathcal{A}$ *we have* $|\Pr[\mathsf{DenGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{DenGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.

Below, we present a definition of symmetric-key encryption with deniable edits that follows the weaker *dual-scheme* paradigm. In this case there are two different encryption schemes: a *default* scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a *denying* scheme $(\mathsf{DenGen}, \mathsf{DenEnc}, \mathsf{DenDec})$. Most users are expected to use the default scheme. However, if a user decides to use the denying scheme instead, she can "deny" the contents of a particular ciphertext c encrypting some message m under sk by producing a secret key $\mathsf{sk}_{c,e} \leftarrow \mathsf{Deny}_{\mathsf{sk}}(c, e)$ that looks like a legitimate key for the default scheme and decrypts c to $m' = \mathsf{Edit}(m, e)$. Even given access to an encryption oracle, a ciphertext c and a key sk, the coercer cannot tell whether (1) all ciphertexts are generated using the default scheme, c is an encryption of m', and sk is the honestly generated key of the default scheme, versus (2) all ciphertexts are generated using the denying scheme, c is an encryption of m and $\mathsf{sk} = \mathsf{sk}_{c,e}$ is the output of the Deny algorithm.

Definition 4.5 (Dual-Scheme Deniable Edits)**.** *A* dual-scheme Edit-deniable symmetric-key encryption scheme *with message-length* $n = n(\lambda)$*, edit description length* $\ell = \ell(\lambda)$*, and edit function* $\mathsf{Edit} : \{0,1\}^{n(\lambda)} \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{n(\lambda)}$ *consists of PPT algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{DenGen}, \mathsf{DenEnc}, \mathsf{DenDec}, \mathsf{Deny})$. *The default scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *and the denying scheme* $(\mathsf{DenGen}, \mathsf{DenEnc}, \mathsf{DenDec})$ *have the usual symmetric-key encryption syntax. The algorithm* $\mathsf{sk}_{c,e} \leftarrow \mathsf{Deny}_{\mathsf{sk}}(c, e)$ *takes as input some ciphertext* c *encrypting data* m *under the denying scheme with secret key* sk*, along with an edit* e *and outputs a secret key* $\mathsf{sk}_{c,e}$ *that decrypts* c *to* $m' = \mathsf{Edit}(m, e)$.

Correctness & Encryption Security. *The schemes* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *and* $(\mathsf{DenGen}, \mathsf{DenEnc}, \mathsf{DenDec})$ *satisfy the standard notions of symmetric-key encryption correctness and CPA security (Definition A.2).*

Deniability. *We define the "deniability game"* $\mathsf{DenGame}^b_{\mathcal{A}}(\lambda)$ *between an adversary* $\mathcal{A}$ *and a challenger with a challenge bit* $b \in \{0,1\}$ *as follows:*

- *If* $b = 0$ *sample* $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$ *and if* $b = 1$ *sample* $\mathsf{sk}^* \leftarrow \mathsf{DenGen}(1^\lambda)$. *Let* $\mathcal{O}(\cdot)$ *be the encryption oracle with* $\mathcal{O}(\cdot) := \mathsf{Enc}_{\mathsf{sk}}(\cdot)$ *if* $b = 0$ *and* $\mathcal{O}(\cdot) := \mathsf{DenEnc}_{\mathsf{sk}^*}(\cdot)$ *if* $b = 1$.
- $\mathcal{A}^{\mathcal{O}}$ *gets access to the encryption oracle and eventually chooses a message* $m \in \{0,1\}^{n(\lambda)}$ *and an edit* $e \in \{0,1\}^{\ell(\lambda)}$.
- *If* $b = 0$*, sample* $c \leftarrow \mathsf{Enc}_{\mathsf{sk}}(\mathsf{Edit}(m, e))$ *and give* (sk, c) *to* $\mathcal{A}$.
  If $b = 1$*, sample* $c \leftarrow \mathsf{DenEnc}_{\mathsf{sk}^*}(m)$, $\mathsf{sk}_{c,e} \leftarrow \mathsf{Deny}_{\mathsf{sk}^*}(c, e)$ *and give* $(\mathsf{sk}_{c,e}, c)$ *to* $\mathcal{A}$.
- $\mathcal{A}^{\mathcal{O}}$ *gets further access to the encryption oracle and eventually outputs a bit* b' *which we define as the output of the game.*

For all PPT adversary $\mathcal{A}$ *we require* $|\Pr[\mathsf{DenGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{DenGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.

4.2.1 Construction

Our construction of dual-scheme symmetric-key encryption with deniable edits follows the same general approach as out public-key construction.

Construction 4.6 (Dual-Scheme Deniable Edit)**.** For any polynomial-time edit function $\mathsf{Edit} : \{0,1\}^{n(\lambda)} \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{n(\lambda)}$ we construct a *dual-scheme* Edit*-deniable symmetric-key encryption* $\mathsf{DSDenE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{DenGen}, \mathsf{DenEnc}, \mathsf{DenDec}, \mathsf{Deny})$, using a single-key symmetric-key functional encryption $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with special encryption and decryption (Definitions 2.3, 2.4) for the function $F := F_{\mathsf{Edit}}$ (Definition 3.2). The construction proceeds as follows.

$\mathsf{DSDenE.DenGen}(1^\lambda)$:

- $\mathsf{msk} \leftarrow \mathsf{FE.Setup}(1^\lambda)$
- Output $\mathsf{sk}^* := \mathsf{msk}$

$\mathsf{DSDenE.Gen}(1^\lambda)$:

- $\mathsf{msk} \leftarrow \mathsf{FE.Setup}(1^\lambda)$
- Select uniform $y \leftarrow \{0,1\}^{\lambda+\ell}$
- $\mathsf{sk}_y \leftarrow \mathsf{FE.Gen}_{\mathsf{msk}}(y)$
- Output $\mathsf{sk} := \mathsf{sk}_y$

$\mathsf{DSDenE.DenEnc}_{\mathsf{sk}^*}(m)$:

- Select uniform $k \leftarrow \{0,1\}^{\lambda+\ell}$
- Output $c \leftarrow \mathsf{FE.Enc}_{\mathsf{msk}}((m,k))$

$\mathsf{DSDenE.Enc}_{\mathsf{sk}}(m)$:

- Select uniform $k \leftarrow \{0,1\}^{\lambda+\ell}$
- Output $c \leftarrow \mathsf{FE.Enc}_{\mathsf{sk}_y}((m,k))$

$\mathsf{DSDenE.DenDec}_{\mathsf{sk}^*}(c)$:

- $(m,k) = \mathsf{FE.Dec}_{\mathsf{msk}}(c)$
- Output m

$\mathsf{DSDenE.Dec}_{\mathsf{sk}}(c)$:

- Output $m = \mathsf{FE.Dec}_{\mathsf{sk}_y}(c)$

$\mathsf{DSDenE.Deny}_{\mathsf{sk}^*}(c, e)$:

- $(m,k) = \mathsf{FE.Dec}_{\mathsf{msk}}(c)$
- $\mathsf{sk}_y \leftarrow \mathsf{FE.Gen}_{\mathsf{msk}}(y)$ where $y = k \oplus (0^\lambda, e)$
- Output $\mathsf{sk}_{c,e} = \mathsf{sk}_y$

Theorem 4.7. *The scheme* DSDenE *given in the above construction 4.6 is a secure* dual-scheme Edit-deniable symmetric-key encryption *if* FE *is a single-key symmetric-key functional encryption with special encryption and decryption for the function* F_{Edit}*. In particular, the construction only relies on the existence of one-way functions.*

Proof. We now prove that Construction 4.6 satisfies the properties of *dual-scheme* Edit*-deniable symmetric-key encryption* in Definition 4.5.

Correctness For every security parameter λ, and message $m \in \{0,1\}^n$:

$$\begin{aligned}
&\Pr\left[m = \mathsf{DSDenE.Dec}_{\mathsf{sk}}(\mathsf{DSDenE.Enc}_{\mathsf{sk}}(m)) \,\middle|\, \mathsf{sk} \leftarrow \mathsf{DSDenE.Gen}(1^\lambda)\right] \\
&= \Pr\left[m = \mathsf{FE.Dec}_{\mathsf{sk}_y}(\mathsf{FE.Enc}_{\mathsf{sk}_y}((m,k))) \,\middle|\, \begin{array}{l} (k,y) \leftarrow \{0,1\}^{\lambda+\ell} \times \{0,1\}^{\lambda+\ell} \\ \mathsf{msk} \leftarrow \mathsf{FE.Setup}(1^\lambda) \\ \mathsf{sk}_y \leftarrow \mathsf{FE.Gen}_{\mathsf{msk}}(y) \end{array}\right] \\
&= \Pr\left[m = F((m,k),y) \,\middle|\, (k,y) \leftarrow \{0,1\}^{\lambda+\ell} \times \{0,1\}^{\lambda+\ell}\right] \\
&= 1 - \Pr\left[\exists e \in \{0,1\}^\ell : y \oplus k = (0^\lambda, e) \,\middle|\, (k,y) \leftarrow \{0,1\}^{\lambda+\ell} \times \{0,1\}^{\lambda+\ell}\right] \\
&= 1 - \frac{1}{2^\lambda}
\end{aligned}$$

Therefore, the scheme $(\mathsf{DSDenE.Gen}, \mathsf{DSDenE.Enc}, \mathsf{DSDenE.Dec})$ is correct. Moreover, for every security parameter λ, and message $m \in \{0,1\}^n$:

$$\begin{aligned}
&\Pr\left[m = \mathsf{DSDenE.DenDec}_{\mathsf{sk}^*}(\mathsf{DSDenE.DenEnc}_{\mathsf{sk}^*}(m)) \,\middle|\, \mathsf{sk}^* \leftarrow \mathsf{DSDenE.DenGen}(1^\lambda)\right] \\
&= \Pr\left[(m,k) = \mathsf{FE.Dec}_{\mathsf{msk}}(\mathsf{FE.Enc}_{\mathsf{msk}}((m,k))) \,\middle|\, \begin{array}{l} k \leftarrow \{0,1\}^{\lambda+\ell} \\ \mathsf{msk} \leftarrow \mathsf{FE.Setup}(1^\lambda) \end{array}\right] \\
&= 1
\end{aligned}$$

Thus, also the scheme $(\mathsf{DSDenE.DenGen}, \mathsf{DSDenE.DenEnc}, \mathsf{DSDenE.DenDec})$ is correct.

Encryption Security. The scheme $(\mathsf{DSDenE.DenGen}, \mathsf{DSDenE.DenEnc}, \mathsf{DSDenE.DenDec})$ is symmetrically secure (i.e., CPA secure). Namely, for every PPT adversary $\mathcal{A}$:

$$|\Pr[\mathsf{CPAGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{CPAGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

This holds because a PPT adversary $\mathcal{A}$ who breaks the CPA security also breaks the single-key symmetric-key functional-encryption security game FEGame (with no secret key, when $y = \perp$).

The scheme $(\mathsf{DSDenE.Gen}, \mathsf{DSDenE.Enc}, \mathsf{DSDenE.Dec})$ is also CPA secure. To prove this we introduce two hybrid games $\mathsf{HybGame}^b_{\mathcal{A}}(\lambda)$ where, instead of using the encryption oracle and the challenge ciphertext defined as follows:

$$\begin{aligned}
\mathcal{O}(\cdot) &:= \mathsf{DSDenE.Enc}(\cdot) = \mathsf{FE.Enc}_{\mathsf{sk}_y}(\cdot, k)_{k \leftarrow \{0,1\}^{\lambda+\ell}} \\
c &\leftarrow \mathsf{DSDenE.Enc}(m_b) = \mathsf{FE.Enc}_{\mathsf{sk}_y}(m_b, k)_{k \leftarrow \{0,1\}^{\lambda+\ell}}
\end{aligned}$$

we replace them with the following modification:

$$\begin{aligned}
\mathcal{O}(\cdot) &:= \mathsf{FE.Enc}_{\mathsf{msk}}(\cdot, k)_{k \leftarrow \{0,1\}^{\lambda+\ell}} \\
c &\leftarrow \mathsf{FE.Enc}_{\mathsf{msk}}(m_b, k)_{k \leftarrow \{0,1\}^{\lambda+\ell}}
\end{aligned}$$

Now we argue that $\mathsf{CPAGame}^0$ is indistinguishable from $\mathsf{HybGame}^0$ which follows by the special encryption property of the FE scheme (Definition 2.3). Furthermore $\mathsf{HybGame}^0$ is indistinguishable from $\mathsf{HybGame}^1$ by the single-key symmetric-key functional-encryption security game FEGame (with no secret key, when $y = \perp$). Lastly $\mathsf{HybGame}^1$ is indistinguishable from $\mathsf{CPAGame}^1$ by the special encryption property.

Deniability. We want to show that for any PPT adversary $\mathcal{A}$:

$$|\Pr[\mathsf{DenGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{DenGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

This follows since an adversary $\mathcal{A}$ who wins the "deniability game" $\mathsf{DenGame}$ with message m and edit e also wins in the single-key symmetric-key functional-encryption security game FEGame with random $y \leftarrow \{0,1\}^{\lambda+\ell}$, and messages $x_0 = (\mathsf{Edit}(m,e), k)$ and $x_1 = (m, k')$ where $k \leftarrow \{0,1\}^{\lambda+\ell}$ and $k' = y \oplus (0^{\lambda+\ell}, e)$. Formally, we prove it by a sequence of hybrids where we change the distribution of the encryption oracle $\mathcal{O}(\cdot)$ and the challenge (c, sk).

Hybrid 0: we starts with the deniability game with $b = 0$, $\mathsf{DenGame}^0_{\mathcal{A}}(\lambda)$. The encryption oracle and the challenge are:

$$\begin{aligned} \mathcal{O}(\cdot) &:= \mathsf{DSDenE.Enc}_{\mathsf{sk}}(\cdot) && = \mathsf{FE.Enc}_{\mathsf{sk}_y}(\cdot, k)_{k \leftarrow \{0,1\}^{\lambda+\ell}} \\ (c, \mathsf{sk}) &:= (\mathsf{DSDenE.Enc}_{\mathsf{sk}}(\mathsf{Edit}(m, e)), \mathsf{sk}) && = (\mathsf{FE.Enc}_{\mathsf{sk}_y}(\mathsf{Edit}(m, e), k'), \mathsf{sk}_y) \end{aligned}$$

and $(y, k') \leftarrow \{0,1\}^{\lambda+\ell} \times \{0,1\}^{\lambda+\ell}$.

Hybrid 1: we move to a hybrid game $\mathsf{HybGame}_{\mathcal{A}}(\lambda)$ (a modification of $\mathsf{DenGame}^0_{\mathcal{A}}(\lambda)$), in which we encrypt using $\mathsf{FE.Enc}_{\mathsf{msk}}$ (instead of using $\mathsf{FE.Enc}_{\mathsf{sk}_y}$). The encryption oracle and the challenge are:

$$\begin{aligned} \mathcal{O}(\cdot) &:= \mathsf{FE.Enc}_{\mathsf{msk}}(\cdot, k)_{k \leftarrow \{0,1\}^{\lambda+\ell}} \\ (c, \mathsf{sk}) &:= (\mathsf{FE.Enc}_{\mathsf{msk}}(\mathsf{Edit}(m, e), k'), \mathsf{sk}_y) \end{aligned}$$

where $(y, k') \leftarrow \{0,1\}^{\lambda+\ell} \times \{0,1\}^{\lambda+\ell}$. By the special encryption property (Definition 2.3) of the underline functional encryption scheme,

$$\mathsf{DenGame}^0_{\mathcal{A}}(\lambda) \overset{c}{\approx} \mathsf{HybGame}_{\mathcal{A}}(\lambda)$$

Hybrid 2: we move to the deniability game with $b = 1$, $\mathsf{DenGame}^1_{\mathcal{A}}(\lambda)$. The encryption oracle and the challenge are:

$$\begin{aligned} \mathcal{O}(\cdot) &:= \mathsf{DSDenE.Enc}_{\mathsf{sk}}(\cdot) && = \mathsf{FE.Enc}_{\mathsf{msk}}(\cdot, k)_{k \leftarrow \{0,1\}^{\lambda+\ell}} \\ (c, \mathsf{sk}) &:= (\mathsf{DSDenE.Enc}_{\mathsf{msk}}(m), \mathsf{DSDenE.Deny}_{\mathsf{msk}}(c, e)) && = (\mathsf{FE.Enc}_{\mathsf{msk}}(m, k'), \mathsf{sk}_y) \end{aligned}$$

where $k' \leftarrow \{0,1\}^{\lambda+\ell}$ and $y := k' \oplus (0^\lambda, e)$. This is equivalent to $y \leftarrow \{0,1\}^{\lambda+\ell}$ and $k' := y \oplus (0^\lambda, e)$. By the security of the functional encryption[4].

$$\mathsf{HybGame}_{\mathcal{A}}(\lambda) \overset{c}{\approx} \mathsf{DenGame}^1_{\mathcal{A}}(\lambda)$$

□

4.3 Efficiency

We note that the efficiency of our public-key and symmetric-key encryption schemes with deniable edits are the same as the analogous constructions of schemes with invisible edits, see Section 3.3.

4.4 Extensions

We now briefly and informally describe two extensions of our deniable schemes.

Bounded-Ciphertext Deniability. Our notion of deniable edits allows us to edit the contents of a single targeted ciphertext c via an edit e by producing a legitimate looking secret key $\mathsf{sk}_{c,e}$. We can also extend our scheme to allowing us to edit the contents of some bounded number of ciphertexts $\vec{c} = (c_1, \ldots, c_t)$ via edits $\vec{e} = (e_1, \ldots, e_t)$ by producing a secret key $\mathsf{sk}_{\vec{c},\vec{e}}$. The construction is essentially the same as before but we use an FE scheme for the function $F(x = (m, k), \vec{y} = (y_1, \ldots, y_t))$ which checks whether there exists some y_i such that $k \oplus y_i = (0^\lambda || e)$ and if so outputs $\mathsf{Edit}(m, e)$ else outputs m. The key generation algorithm would output an FE secret key $\mathsf{sk}_{\vec{y}}$ for a uniformly random vector $\vec{y}$. To deny a vector of ciphertexts $\vec{c} = (c_1, \ldots, c_t)$ where each c_i is an FE encryption of $x_i = (m_i, k_i)$ we would create an FE secret key for the vector $\vec{y} = (y_1, \ldots, y_t)$ where $y_i = (0^\lambda || e_i) \oplus k_i$. The cost of this construction is that now the secret key size scales with $\ell \cdot t$ where ℓ is the edit description size and t is the number of ciphertexts.

We could also consider yet another variant where we want to edit the contents of t ciphertexts $\vec{c} = (c_1, \ldots, c_t)$ via a single edit e to be applied to all of them by creating a secret key $\mathsf{sk}_{\vec{c},e}$. In that case we could

[4] FE game with a random y and messages $m_0 = (m, y \oplus (0^\lambda, e))$, $m_1 = (\mathsf{Edit}(m, e), k')$ for a random k'.

use an FE scheme for the function $F(x = (m, k), (y^*, y_1, \ldots, y_t))$ which checks if $k \oplus (y_i || y^*) = (0^\lambda || e)$ and if so output $\mathsf{Edit}(m, e)$ else m. Otherwise the scheme would be analogous to the previous one. The allows us to get a construction where the secret key size scales with $\ell + t$ instead of $\ell \cdot t$.

Later, when we consider invisible edits, we will be able to edit the contents of an unbounded number of ciphertexts via an edit e by generating a secret key sk_e of some fixed size. However, in that case we will also need to be able to also plausibly lie about the public key by giving out a modified public key pk_e instead of pk. This is in contrast to bounded-ciphertext deniability discussed above where the secret key $\mathsf{sk}_{\vec{c},e}$ looks like a legitimate secret key for the original public key pk.

Denying Randomness of Key Generation. For simplicity, we previously assumed that the coercing adversary can only ask the user for her secret key but not for the randomness of the key generation algorithm which the user can plausibly delete. However, it would be relatively easy to also allow the user to deniably generate fake randomness for the key generation algorithm as well. We briefly sketch this extension.

Let's start by considering this extension in the public-key setting. The way we defined the syntax of deniable public-key encryption, we had a single key generation algorithm that outputs $(\mathsf{pk}, \mathsf{sk}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda)$. In order for the above extension to be possible we must now consider two algorithms (moving us closer to the two-scheme rather than two-key setting), a default one that outputs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and a deniable one that outputs $(\mathsf{pk}, \mathsf{sk}, \mathsf{dk}) \leftarrow \mathsf{DenGen}(1^\lambda)$. Given dk, c, e where $c = \mathsf{Enc}_{\mathsf{pk}}(m)$, the denying procedure should now output $r \leftarrow \mathsf{Deny}_{\mathsf{dk}}(c, e)$ such that if we run $\mathsf{sk}_{c,e} = \mathsf{Gen}(1^\lambda; r)$ then $\mathsf{Dec}_{\mathsf{sk}_{c,e}}(c) = \mathsf{Edit}(m, e)$. Security is defined analogously except that the adversary gets the key generation randomness r rather than sk.

It turns out that our construction (Construction 4.2) already essentially achieves this if we start with a *simulatable Functional Encryption* scheme where it's possible to obliviously generate $(\mathsf{mpk}, \mathsf{sk}_y) = \mathsf{OGen}(1^\lambda, y; r)$ without knowing msk so that even given the randomness r one cannot distinguish encryptions of x_0, x_1 if $F(x_0, y) = F(x_1, y)$. Moreover given $\mathsf{mpk}, \mathsf{msk}$ and sk_y it's possible to come up with legitimate looking random coins r such that $(\mathsf{mpk}, \mathsf{sk}_y) \leftarrow \mathsf{OGen}(1^\lambda; r)$. Using this type of FE we can modify the $\mathsf{DEdit.Gen}$ algorithm to run $\mathsf{FE.OGen}$. Using $\mathsf{dk} = \mathsf{msk}$ we'd then be able to sample legitimate looking random coins for OGen.

It furthermore turns out that the construction of FE from PKE already gives us the simulatable FE property if we start with a simulatable PKE [DN00].

The same idea also works in the symmetric key setting analogously, using only one-way functions.

Acknowledgment

Thanks to Omer Paneth for interesting discussions on the topics of this work.

References

[BNNO11] Rikke Bendlin, Jesper Buus Nielsen, Peter Sebastian Nordholt, and Claudio Orlandi. Lower and upper bounds for deniable public-key encryption. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 125–142, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany.

[CDMW09] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Improved non-committing encryption with applications to adaptively secure protocols. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 287–302, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany.

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 90–104, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Heidelberg, Germany.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on Theory of Computing*, pages 639–648, Philadephia, PA, USA, May 22–24, 1996. ACM Press.

[CPV16] Ran Canetti, Oxana Poburinnaya, and Muthuramakrishnan Venkitasubramaniam. Equivocating yao: Constant-round adaptively secure multiparty computation in the plain model. Cryptology ePrint Archive, Report 2016/1190, 2016. `http://eprint.iacr.org/2016/1190`.

[DIO16] Angelo De Caro, Vincenzo Iovino, and Adam O'Neill. Deniable functional encryption. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 9614 of *Lecture Notes in Computer Science*, pages 196–222, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany.

[DN00] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 432–450, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Heidelberg, Germany.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 162–179, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

[GWZ09] Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 505–523, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany.

[HJO$^+$16] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 149–178, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany.

[OPW11] Adam O'Neill, Chris Peikert, and Brent Waters. Bi-deniable public-key encryption. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 525–542, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10: 17th Conference on Computer and Communications Security*, pages 463–472, Chicago, Illinois, USA, October 4–8, 2010. ACM Press.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

A Standard Cryptographic Definitions

A.1 Encryption Scheme Definitions

Definition A.1 (Public-Key Encryption)**.** *A* public-key encryption *consists of PPT algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following syntax:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
- $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$ *outputs an encryption of* m.
- $m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$ *outputs a message* m.

The scheme should satisfy the following properties:

Correctness: *For every security parameter* λ *and message* m*,*

$$\Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m)) = m | (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)] = 1 - \mathsf{negl}(\lambda)$$

CPA Security. *We define the "CPA game"* $\mathsf{CPAGame}^b_{\mathcal{A}}(\lambda)$ *between an adversary* $\mathcal{A}$ *and a challenger with a challenge bit* $b\{0, 1\}$ *as follows:*

- *Sample* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *and give* pk *to* $\mathcal{A}$.
- *The adversary* $\mathcal{A}$ *chooses two messages* $m_0, m_1 \in \{0, 1\}^n$
- *The adversary* $\mathcal{A}$ *gets a challenge* $c \leftarrow \mathsf{Enc}_{\mathsf{sk}}(m_b)$*, and eventually outputs a bit* b' *which we define as the output of the game.*

We require that for all PPT adversary $\mathcal{A}$ *we have* $|\Pr[\mathsf{CPAGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{CPAGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.

Definition A.2 (Symmetric-Key Encryption)**.** *A* symmetric-key encryption *consists of PPT algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following syntax:*

- $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$.
- $c \leftarrow \mathsf{Enc}_{\mathsf{sk}}(m)$ *outputs an encryption of* m.
- $m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$ *outputs a message* m.

The scheme should satisfy the following properties:

Correctness: *For every security parameter* λ *and message* m*,*

$$\Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{sk}}(m)) = m | \mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)] = 1 - \mathsf{negl}(\lambda)$$

CPA Security. *We define the "CPA game"* $\mathsf{CPAGame}^b_{\mathcal{A}}(\lambda)$ *between an adversary* $\mathcal{A}$ *and a challenger with a challenge bit* $b\{0, 1\}$ *as follows:*

- *Sample* $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$.
- *The adversary* $\mathcal{A}^{\mathsf{Enc}_{\mathsf{sk}}(\cdot)}$ *gets the encryption oracle. Eventually, it chooses two messages* $m_0, m_1 \in \{0, 1\}^n$
- *The adversary* $\mathcal{A}$ *gets a challenge* $c \leftarrow \mathsf{Enc}_{\mathsf{sk}}(m_b)$ *and further access to the encryption oracle* $\mathcal{A}^{\mathsf{Enc}_{\mathsf{sk}}(\cdot)}(c)$*, and eventually output a bit* b' *which we define as the output of the game.*

We require that for all PPT adversary $\mathcal{A}$ *we have* $|\Pr[\mathsf{CPAGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{CPAGame}^1_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.

A.2 Garbled Circuits

Definition A.3 (Garbling)**.** *A* garbling scheme *for a class of circuits* $\mathcal{C} = \{C : \{0,1\}^{n_1(\lambda)} \to \{0,1\}^{n_2(\lambda)}\}$ *consists of PPT algorithms* $(\mathsf{Setup}, \mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ *with the following syntax:*

- $\mathbf{K} \leftarrow \mathsf{Setup}(1^\lambda)$ *generates* $\mathbf{K} = \{k_{i,b} : i \in [n_1(\lambda)], b \in \{0,1\}\}$
- $\hat{C} \leftarrow \mathsf{Garble}(C, \mathbf{K})$ *outputs a garbled circuit of the circuit* C*.*
- $\mathbf{K}_x \leftarrow \mathsf{Encode}(x, \mathbf{K})$ *outputs an encoding of the input* $x \in \{0,1\}^{n_1(\lambda)}$, $\mathbf{K}_x = \{k_{i,x[i]}\}_{i \in [n_1(\lambda)]}$ *where* $x[i]$ *is the i-th bit of* x *.*
- $C(x) = \mathsf{Eval}(\hat{C}, \mathbf{K}_x)$ *outputs* $C(x) \in \{0,1\}^{n_2(\lambda)}$*.*

The scheme should satisfy the following properties:

Correctness *For every security parameter* λ*, circuit* $C : \{0,1\}^{n_1(\lambda)} \to \{0,1\}^{n_2(\lambda)}$*, and input* $x \in \{0,1\}^{n_1(\lambda)}$*:*

$$\Pr\left[C(x) = \mathsf{Eval}(\hat{C}, \mathbf{K}_x) \;\middle|\; \begin{array}{l} \mathbf{K} \leftarrow \mathsf{Setup}(1^\lambda) \\ \hat{C} \leftarrow \mathsf{Garble}(C, \mathbf{K}) \\ \mathbf{K}_x \leftarrow \mathsf{Encode}(x, \mathbf{K}) \end{array} \right] = 1.$$

Security *For every security parameter* λ*, circuit* $C : \{0,1\}^{n_1(\lambda)} \to \{0,1\}^{n_2(\lambda)}$*, and pair of inputs* $x_0, x_1 \in \{0,1\}^{n_1(\lambda)}$ *such that* $C(x_0) = C(x_1)$*:*

$$\left\{ \hat{C}, \mathbf{K}_{x_0} | \mathbf{K}_{x_0} \leftarrow \mathsf{Encode}(x_0, \mathbf{K}) \right\} \overset{c}{\approx} \left\{ \hat{C}, \mathbf{K}_{x_1} | \mathbf{K}_{x_1} \leftarrow \mathsf{Encode}(x_1, \mathbf{K}) \right\}$$

where $\hat{C} \leftarrow \mathsf{Garble}(C, \mathbf{K})$ *and* $\mathbf{K} \leftarrow \mathsf{Setup}(1^\lambda)$*.*

B Constructions of Single-Key Functional-Encryption Schemes

Notation: Garbling Two-Input Circuits. It will be useful for us to define some notation for garbling circuits $C(x, y)$ that take two inputs. Let $\mathsf{GC} = (\mathsf{Setup}, \mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ be a garbling scheme for a class of circuits $\mathcal{C} = \{C : \{0,1\}^{n_1(\lambda)} \times \{0,1\}^{n_2(\lambda)} \to \{0,1\}^{n_3(\lambda)}\}$, and let $\mathbf{K} \leftarrow \mathsf{GC.Setup}(1^\lambda)$. Recall $\mathbf{K} = \{k_{i,b} : i \in [n_1 + n_2], b \in \{0,1\}\}$.

We denote by $k^1_{i,b} = k_{i,b}$ for every $i \in [n_1(\lambda)]$, by $k^2_{i,b} = k_{(n_1(\lambda)+i),b}$ for every $i \in [n_2(\lambda)]$, by $\mathbf{K}^1$ the sets of keys for the first input, and by $\mathbf{K}^2$ the sets of keys for the second input i.e.,

$$\begin{aligned} \mathbf{K}^1 &= \{k^1_{i,b} : i \in [n_1(\lambda)], b \in \{0,1\}\} = \{k_{i,b} : i \in [n_1(\lambda)], b \in \{0,1\}\} \\ \mathbf{K}^2 &= \{k^2_{i,b} : i \in [n_2(\lambda)], b \in \{0,1\}\} = \{k_{(n_1(\lambda)+i),b} : i \in [n_2(\lambda)], b \in \{0,1\}\}. \end{aligned}$$

Moreover, if $(\mathbf{K}^1, \mathbf{K}^2) \leftarrow \mathsf{GC.Setup}(1^\lambda)$, we also define notation for encoding part of the input $\mathbf{K}^1_x \leftarrow \mathsf{GC.Encode}(x, \mathbf{K}^1)$ and similarly $\mathbf{K}^2_y \leftarrow \mathsf{GC.Encode}(y, \mathbf{K}^2)$.

B.1 Single-Key Public-Key Functional-Encryption Construction

Construction B.1 (Single-Key PK FE)**.** We construct $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ a *single-key public-key functional-encryption* for a function $F : \{0,1\}^{n_1(\lambda)} \times \{0,1\}^{n_2(\lambda)} \to \{0,1\}^{n_3(\lambda)}$ that has a special decryption algorithm using $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ a public-key encryption scheme and $\mathsf{GC} = (\mathsf{Setup}, \mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ a garbling scheme for a class of circuits $\mathcal{C} = \{C : \{0,1\}^{n_1(\lambda)+n_2(\lambda)} \to \{0,1\}^{n_3(\lambda)}\}$.

We denote by $n_1 = n_1(\lambda), n_2 = n_2(\lambda)$ and $n_3 = n_3(\lambda)$, and abuse notation and denote by F the circuit that compute the function F.

- $\mathsf{FE.Setup}(1^\lambda)$:
  - For $i \in [n_2], b \in \{0,1\}$: select $(\mathsf{sk}_{i,b}, \mathsf{pk}_{i,b}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$.

- Output $(\mathsf{mpk} := \{\mathsf{pk}_{i,b} : i \in [n_2], b \in \{0,1\}\}, \mathsf{msk} := \{\mathsf{sk}_{i,b} : i \in [n_2], b \in \{0,1\}\}.)$

- $\mathsf{FE.Gen}_{\mathsf{msk}}(y)$:
  - Output $\mathsf{sk}_y := \{\mathsf{sk}_{i,y[i]} : i \in [n_2]\}$.
- $\mathsf{FE.Enc}_{\mathsf{mpk}}(x)$:
  - $(\mathbf{K}^1, \mathbf{K}^2) \leftarrow \mathsf{GC.Setup}(1^\lambda)$.
  - $\hat{F} \leftarrow \mathsf{GC.Garble}(F, (\mathbf{K}^1, \mathbf{K}^2))$.
  - $\mathbf{K}^1_x \leftarrow \mathsf{GC.Encode}(x, \mathbf{K}^1)$.
  - For $i \in [n_2], b \in \{0,1\}$: select $c_{i,b} \leftarrow \mathsf{PKE.Enc}_{\mathsf{pk}_{i,b}}(k^2_{i,b})$.
  - Output $c = (\hat{F}, \mathbf{K}^1_x, \{c_{i,b} : i \in [n_2], b \in \{0,1\}\})$.
- $\mathsf{FE.Dec}_{\mathsf{sk}_y}(c)$:
  - For every $i \in [n_2] : k^2_{i,y[i]} = \mathsf{PKE.Dec}_{\mathsf{sk}_{i,y[i]}}(c_{i,y[i]})$.
  - $\mathbf{K}^2_y = \{k^2_{i,y[i]} : i \in [n_2]\}$.
  - Output $F(x,y) = \mathsf{GC.Eval}(\hat{F}, (\mathbf{K}^1_x, \mathbf{K}^2_y))$.

The correctness and security (as in Definition 2.1) of Construction B.1 follows directly from the security of the garbling and public-key encryption schemes.[5] Additionally, a simple modification for the construction above results a scheme with a special decryption algorithm. We change the encryption algorithm to garble the circuit F_{special} (instead of F) where $F_{\mathsf{special}} : \{0,1\}^{n_1} \times \{0,1\}^{n_2+1} \to \{0,1\}^{\max\{n_1,n_3\}}$,

$$F_{\mathsf{special}}(x,(y,b)) := \begin{cases} x & \text{If } (y,b) = (0^{n_2}, 0) \\ F(x,y) & \text{otherwise} \end{cases}$$

To make the output length the same in both cases, pad with zeros.

Using an IBE scheme We change the construction above to work with an *identity based encryption* scheme $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ to reduce the master public-key size mpk. In the resulting scheme the master public-key is of the same size as the master public-key in the IBE scheme.

Instead of generating for every $(i,b) \in [n_2] \times \{0,1\}$ a pair of public-key and secret-key, the setup algorithm generates a pair of master public-key and master secret-key for an IBE scheme, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$. In addition, the key generating algorithm generates n_2 private-keys of the IBE scheme, namely for each identity $(i, y[i])_{i \in [n_2]}$ generates an IBE private-key, $\mathsf{sk}_{i,y[i]} \leftarrow \mathsf{IBE.Gen}_{\mathsf{msk}}(\mathsf{id} = (i, y[i]))$. Further, the FE encryption algorithm now samples $c_{i,b} \leftarrow \mathsf{IBE.Enc}_{\mathsf{mpk}}(\mathsf{id} = (i,b), k^2_{i,b})$ using the IBE encryption algorithm.

B.2 Single-Key Symmetric-Key Functional-Encryption Construction

Construction B.2 (Single-Key SK FE)**.** We construct $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ a *single-key symmetric-key functional-encryption* for a function $F : \{0,1\}^{n_1(\lambda)} \times \{0,1\}^{n_2(\lambda)} \to \{0,1\}^{n_3(\lambda)}$ that has a special encryption and decryption algorithm using $\mathsf{Sym} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ a symmetric-key encryption scheme with pseudorandom ciphertexts and $\mathsf{GC} = (\mathsf{Setup}, \mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval})$ a garbling scheme for a class of circuits $\mathcal{C} = \{C : \{0,1\}^{n_1(\lambda)+n_2(\lambda)} \to \{0,1\}^{n_3(\lambda)}\}$.

We denote by $n_1 = n_1(\lambda), n_2 = n_2(\lambda)$ and $n_3 = n_3(\lambda)$, and abuse notation and denote by F the circuit that compute the function F.

- $\mathsf{FE.Setup}(1^\lambda)$:
  - For $i \in [n_2], b \in \{0,1\}$: select $(\mathsf{sk}_{i,b}) \leftarrow \mathsf{Sym.Gen}(1^\lambda)$.

[5] If we assume perfect correctness of the underline GC and PKE, also the constructed FE has a perfect correctness (as in Definition 2.1).

  - Output ($\mathsf{msk} := \{\mathsf{sk}_{i,b} : i \in [n_2], b \in \{0,1\}\}$.)

- $\mathsf{FE.Gen}_{\mathsf{msk}}(y)$:
  - Output $\mathsf{sk}_y := \{\mathsf{sk}_{i,y[i]} : i \in [n_2]\}$.

- $\mathsf{FE.Enc}_{\mathsf{msk}}(x)$:
  - $(\mathbf{K}^1, \mathbf{K}^2) \leftarrow \mathsf{GC.Setup}(1^\lambda)$.
  - $\hat{F} \leftarrow \mathsf{GC.Garble}(F, (\mathbf{K}^1, \mathbf{K}^2))$.
  - $\mathbf{K}^1_x \leftarrow \mathsf{GC.Encode}(x, \mathbf{K}^1)$.
  - For $i \in [n_2], b \in \{0,1\}$: select $c_{i,b} \leftarrow \mathsf{PKE.Enc}_{\mathsf{sk}_{i,b}}(k^2_{i,b})$.
  - Output $c = (\hat{F}, \mathbf{K}^1_x, \{c_{i,b} : i \in [n_2], b \in \{0,1\}\})$.

- $\mathsf{FE.Dec}_{\mathsf{sk}_y}(c)$:
  - For every $i \in [n_2] : k^2_{i,y[i]} = \mathsf{PKE.Dec}_{\mathsf{sk}_{i,y[i]}}(c_{i,y[i]})$.
  - $\mathbf{K}^2_y = \{k^2_{i,y[i]} : i \in [n_2]\}$.
  - Output $F(x,y) = \mathsf{GC.Eval}(\hat{F}, (\mathbf{K}^1_x, \mathbf{K}^2_y))$.

The correctness and security (as in Definition 2.2) of Construction B.2 follows directly from the security of the garbling and symmetric-key encryption schemes.[5] If ciphertexts of the underline symmetric-key encryption are computationally indistinguishable from random strings, then we also get a symmetric-key functional encryption with a special encryption algorithm. In particular, to encrypt with sk_y, the encryption algorithm outputs uniformly random strings in place of $c_{i,1-y[i]} \leftarrow \mathsf{Sym.Enc}_{\mathsf{sk}_{i,(1-y[i])}}[k^2_{i,(1-y[i])}]$ for every $i \in [n_2]$. Additionally, we cam use the same trick we did in the public-key case to get the special decryption property.