Mergeable Functional Encryption

Vincenzo Iovino[1] Karol Żebrowski[2]

[1] University of Luxembourg, vincenzo.iovino@uni.lu
[2] University of Warsaw, kz277580@students.mimuw.edu.pl

Abstract

In recent years, there has been great interest in Functional Encryption (FE), a generalization of traditional encryption where a token enables a user to learn a specific function of the encrypted data and nothing else. In this paper we put forward a new generalization of FE that we call Mergeable FE (mFE). In a mFE system, given a ciphertext c_1 encrypting m_1 and a ciphertext c_2 encrypting m_2, it is possible to produce in an oblivious way (i.e., given only the public-key and without knowledge of the messages, master secret-key or any other auxiliary information) a ciphertext encrypting the string $m_1||m_2$ under the security constraint that this new ciphertext does not leak more information about the original messages than what may be leaked from the new ciphertext using the tokens. For instance, suppose that the adversary is given the token for the function $f(\cdot)$ defined so that for strings $x \in \{0,1\}^n$, $f(x) \overset{\triangle}{=} g(x)$ for some function $g : \{0,1\}^n \to \{0,1\}$ and for strings $y = (x_1||x_2) \in \{0,1\}^{2n}$, $f(x_1||x_2) \overset{\triangle}{=} g(x_1) \vee g(x_2)$. Furthermore, suppose that the adversary gets a ciphertext c encrypting $(x_1||x_2)$ that is the result of "merging" some ciphertexts c_1 and c_2 encrypting respectively x_1 and x_2, and suppose that the token for f evaluates to 1 on c. Then, the security of mFE guarantees that the adversary only learns the output $f(x_1, x_2) \overset{\triangle}{=} g(x_1) \vee g(x_2) = 1$ and nothing else (e.g., the adversary should not learn whether $g(x_1) = 1$ or $g(x_2) = 1$). This primitive is in some sense FE with the "best possible" homomorphic properties and, besides being interesting in itself, it offers wide applications. For instance, it has as special case multi-inputs FE and thus indistinguishability obfuscation (iO) and extends the latter to support more efficiently *homomorphic* and *re-randomizable* properties. We construct mFE schemes supporting a single merging operation, one from indistinguishability obfuscation for Turing machines and one for messages of unbounded length from *public-coin* differing-inputs obfuscation. Finally, we discuss a construction supporting unbounded merging operations from new assumptions.

Keywords: Functional Encryption, Obfuscation, Homomorphic cryptography.

Contents

1 Introduction

Functional Encryption (FE) [BSW11] is a sophisticated type of encryption that allows to finely control the amount of information that is revealed by a ciphertext. In a FE scheme, for any function f allowed by the system, the owner of the master secret key can compute a restricted key, called *token*, for f, that enables to compute $f(m)$ on a ciphertext encrypting m, and nothing else. In recent years, more expressive forms of FE were constructed in a series of works (see, e.g., [BDOP04, BW07, KSW08, LOS$^+$10, OT12, Wat12]) culminating in the breakthrough of Garg *et al.* [GGH$^+$13] who showed the first candidate construction of FE for all polynomial-size circuits from indistinguishability obfuscation. Another line of research investigated extensions and generalizations of FE such as multi-inputs FE [GGG$^+$14, BLR$^+$14], FE for randomized functionalities [GJKS13, KSY14], FE in the private-key model [SSW09, BS14] and in alternative models [BRS13a, BRS13b, BIP10]. While these works offer unique applications and pose new insights and challenges, we call for the need for a new and further generalization of FE not previously discussed in the literature.

1.1 Mergeable functional encryption.

We put forward the concept of mergeable FE (mFE) scheme. A mFE scheme is identical to a FE scheme but in addition it is endowed with a Merge algorithm that given two ciphertexts c_1 and c_2 encrypting respectively m_1 and m_2 can produce a ciphertext c encrypting $m_1||m_2$ where "'$||$ represents the concatenation symbol, i.e., can *merge* the original ciphertexts, in an oblivious way without knowledge of the underlying plaintexts. As special case, a mFE also allows to *update* a ciphertext in an oblivious way. That is, having a ciphertext encrypting an unknown plaintext m, the system allows to produce a ciphertext c encrypting $m||m_2$ where m_2 is any plaintext. Notice that a mFE system represents in some sense a FE scheme with the best possible *homomorphic* properties. Indeed, it is easy to see that a FE scheme can not be in general fully homomorphic: for instance, the token for the function $f(\cdot)$ such that $f(x) = f(y)$ for some messages x, y but $f(g(x)) \neq f(g(y))$ for some function $g(\cdot)$, allows to distinguish whether a ciphertext c encrypts x or y by homomorphically evaluating the function $g(\cdot)$ on c. Instead, the restricted form of homomorphism allowed by mFE preserves and does *not* contradict its functional properties. Apart from being interesting in itself, the applications of mFE and the settings where it can be applied are vast and we will illustrate few of them.

1.2 Applications

mFE implies Multi-inputs FE. The works of [GGG$^+$14, GKL$^+$13, GGJS13] introduced the concept of multi-inputs FE (MI-FE), a generalization of FE where a token corresponds to a multi-variate function that takes multiple ciphertexts as input. In these works, several settings were defined. mFE implies MI-FE[1]: to evaluate a multi-variate token on multiple ciphertexts it is sufficient to merge the ciphertexts. Furthermore, mFE generalizes previous works on MI-FE to functions with *unbounded arity*, i.e., not putting any bound on the arity of the allowed functions. We stress that as special case mFE ssupporting a *single* merging operation implies 2-inputs MI-FE.

mFE implies efficient homomorphic, updatable and re-randomizable indistinguishability obfuscation. Gordon *et al.* [GGG$^+$14] show that MI-FE (specifically, 2-inputs FE) im-

[1]This holds for the public-key setting where the adversary is given the public-key that allows to encrypt messages corresponding to any dimension

plies indistinguishability obfuscation (iO) [GGH$^+$13] that thus is implied by mFE. In addition, it is easy to see that mFE has as byproduct a form of homomorphic obfuscation in the CRS model where given the obfuscations of two programs $P_1(\cdot)$ and $P_2(\cdot)$ computed with respect to the same CRS and any program $Q(\cdot)$, it is possible to derive an obfuscation of the program $P_3(x) \stackrel{\triangle}{=} Q(P_1(x), P_2(x))$. If, in addition the program P_2 is given in clear (i.e., non obfuscated), the CRS is not needed and we call this primitive *updatable* obfuscation. These implications follow from the fact that a program P can be obfuscated with mFE by outputting its encryption Ct, the public-key, and the token Tok for the universal circuit $U(P, x) \stackrel{\triangle}{=} P(x)$, and can be evaluated on any input x by just merging Ct with the encryption of x (computed with the public-key) and evaluating the resulting ciphertext with Tok. Homomorphic obfuscation can be put in the construction of FE of Garg *et al.* [GGH$^+$13] to enable a deleagation feature, specifically the capability of deriving from a token for a function f a token for a more restricted function f'. A similar system was proposed by Boneh *et al.* [BGG$^+$14] in the context of attribute-based encryption, a restricted form of FE. Moreover, such system allows to *merge* two tokens to produce a token for their composition, a feature not shared by the aforementioned work of Boneh *et al.*Homomorphic obfuscation also implies a form of *re-randomizable iO* (riO) (without assuming the CRS model) where an obfuscated program can be re-randomized as follows: to re-randomize an obfuscated program P, just derive an iO for the program $P \vee Q$ where Q is the program that outputs 0 on any input. A riO can be used for example by a programmer to produce a new obfuscated program P_2 that is functionally equivalent to P_1 (but without knowledge of the original program P_1) passing P_2 off as a different version of P_1.

We stress that the above primitives are implied directly *already* from iO but less efficiently: indeed building homomorphic iO from iO in the "direct" way results in obfuscations of obfuscations and thus can be less efficient than the solution based on mFE.

mFE implies CCA1-secure PKE with homomorphic properties. It is well known that a PKE scheme can not be fully homomorphic (see also Prabhakaran and Rosulek [PR08] for alternative models). Notwithstanding it is reasonable to ask whether it can be CCA1-secure. Loft *et al.* [LMSV12] present a construction of somewhat homomorphic encryption from special knowledge assumptions. From mFE it is possible to derive CCA1-secure PKE with special homomorphic properties. Recall that FE (specifically, identity-based encryption) implies CCA1-secure public-key encryption (PKE) [CHK04, BCHK07]. If the underlying FE scheme used in the construction of CCA1-secure PKE is in particular a mFE scheme, the resulting PKE scheme would be a CCA1-secure PKE scheme supporting merging operations. Specifically, similarly to the transformation of Canetti *et al.* [CHK04], the CCA1 encryption of message m consists of a mFE ciphertext that encrypts (id, m) for a random identity id. mFE allows to merge such a ciphertext with others without the knowledge of the underlying message m.

Applications to searching over encrypted databases in a public-key setting. One of the most notable applications of FE is to searching over encrypted databases in a cloud computing setting. In this scenario, Alice, the manager of a company, can distribute a token for a function f to any of her employees who can use such tokens to perform queries over encrypted databases located in a cloud server. For instance, one database D_1 is produced by Bob and sent to the cloud server in an encrypted form under the public-key of Alice. Similarly, Eve produced her database D_2 and sent it to the cloud server in the same encrypted form. The two databases could contain information about products sold, respectively, by the companies of Bob and Eve and of interest for the company of Alice. Moreover, we assume that the cloud servers own a lot of computational power and space but are not trusted by Bob and Eve, i.e., Bob and Eve wish to leak as few information as possible to the servers. For simplicity, suppose that the databases are implemented as lists of elements, let us say $D_1 = (x_1||\ldots||x_n)$ and $D_2 = (y_1||\ldots||y_n)$. An employee of the company of Alice could be interested in searching whether a specific product x is in *one* of the two databases but he does not care in which one it is in. Thus, the employee sends to both servers a token for the function $f_x(D) \overset{\triangle}{=} 1$ iff the list D contains x. The server evaluates the token on *both* encrypted databases one at time and then communicate to the employee whether the requested product x is or is not in the databases. Suppose that at some point there is a commercial agreement between the companies of Bob and Eve and as result of it they decide to merge their respective data without compromising the needs of the company of Alice and to keep on storing the merged encrypted data in the same cloud server. One solution could be to reveal to each other their data and re-encrypt anything under the public-key of Alice. However, this is not a valid solution as they wish to preserve the confidentiality of their own data. Another approach could be to store on the server just the concatenation of the previous ciphertexts. That is, if c_1 is the encryption of D_1 and c_2 is the encryption of c_2, then the new encrypted data could just consist of $c_1||c_2$. Anyway, recall that Bob and Eve wish to hide to the cloud server the contents of the new encrypted database. If the new encrypted database was just the concatenation of the encryptions of D_1 and D_2, then from a query for a product x, the server could figure out whether x was in the database of Alice or in the database of Bob by running the token separately on c_1 and c_2. Thus, this solution is not satisfactory. Another approach could to make Alice to create a *new* FE system and to ask the server to merge the two encrypted data under the *new* Alice's public-key. This solution incurs in a lot of problems as well. First of all, it requires a work from the Alice's side. Instead, Bob and Eve would like to merge their data without involving Alice's company (after all, recall that the employees of Alice are interested in searching whether a specific product is or is not in *one* of the encrypted databases and *not* also on which one it is in). That is, in the scenario we envision, Alice consents the companies with which she collaborates with (using their encrypted data) to merge their own encrypted data without even informing her, i.e., in a *non-interactive* way. Furthermore, the size of the public-keys and parameters would grow (as the new system is based on the old one) and the main drawback is that Alice would have to re-compute and re-distribute *new* tokens to each employee. Instead, mFE offers a valid solution to this problem: if the databases are encrypted with a mFE system, then Bob and Eve can merge their own encrypted databases in an oblivious way hiding any information on the original databases to the server.

Applications to updating encrypted data in a private-key setting. In a private-key setting, Alice delegates her encrypted data D to a powerful cloud server and at any point she can perform a search or any computation on the data sending to the server a token for the desired function. Moreover, Alice can compute this token from a device with a low computational power as a mobile phone in which the original data is *not* present. Suppose now that Alice needs to add a file x to her encrypted data. That is, she wishes to update her encrypted data so that it now should encrypts $D||x$. As before, Alice does not consider satisfactory a solution where a new ciphertext encrypting x is added in the server. For concreteness, suppose that Alice wishes to compute the following function: $f(D||x) = g(D) \vee g(x)$ on the encrypted data and suppose that we adopt the trivial approach in which the server stores the two ciphertexts, and suppose that Alice sends to the server the two tokens, one for $g(D)$ and one for $g(x)$. Then, the server would learn whether $g(D) = 1$ or $g(x) = 1$ whereas Alice wishes the server to only learn $f(D||x)$ and nothing else. mFE offers a valid solution to this problem: Alice could send the encryption of the new data to the server asking the server to merge it with the old encrypted data. One could object that the solutions is not satisfactory because the server could keep on storing *both* ciphertexts, the one encrypting D and the one encrypting x in addition to the one encrypting $D||x$, so to be able to leak the undesired information (e.g., $g(D)$ and $g(x)$). This problem is easily fixed in the following way. Alice can update on the server the encrypted data $\$x$, where $\$$ is a special symbol not present in D or x. The functions f's for which she will compute tokens afterwards will be defined so to check (1) whether the input contains the special symbol in the middle (i.e., the functions will check whether the input y has the form $D\$x$ for some strings D and x) and (2) for inputs not satisfying this condition, the function is defined to output an error $\perp$. In this way, the server can only use the new tokens on the updated encrypted data and not on the previous ones.

Applications to verifiable computation. Using the techniques of Gennaro *et al.* [GGP10], it is also possible to add verifiability to the above scenario so that Alice can check whether the server correctly computed the function. Thus, mFE allows to extend non-interactive verifiable computation to a setting where different untrusted workers have the capability to merge their respective encrypted data. In this scenario, suppose that Alice has delegated two different data, D_1 and D_2, to two different servers. Moreover, suppose that Alice has completely deleted the original data that reside only on the servers in encrypted form. mFE allows Alice to request to the servers to merge their encrypted data in an oblivious way preserving the privacy of the original data and limiting the amount of information leaked by future queries. Moreover, in the case of only one worker, mFE allows Alice to *update* the encrypted data on the server with the same security guarantees.

1.3 mFE: properties and security

The requirements of a mFE scheme. In view of the above and further applications we desire mFE systems to satisfy the following properties:

- The operation of merging two ciphertexts c_1 and c_2 can be performed having just the public-key of the system, c_1 and c_2.
- The size of the merged ciphertext should be proportional to the sum of the lengths of the old ciphertexts plus an *additive* factor polynomial in the security parameter. That is, suppose that c_1 has size m_1 and c_2 has size m_2 and let k be the security parameter. Then, the the result of merging c_1 and c_2 must be a ciphertext of length $O(m_1 + m_2 + k)$. This is to rule out solutions where the ciphertext resulting from the merge has for instance length $k \cdot (|m_1| + \cdot|m_2|)$ that would bound the number of mergings to be logarithmic (or less) in the security parameter. We call this requirement *compactness*.
- The systems must be designed for the Turing Machine (TM) model of computation as the circuit model does not fit well with the mFE setting. In fact, circuits can compute over a fixed number of bits, and thus, even though the system allowed multiple merging operations, a token could be used only on ciphertexts resulting from a bounded number of merging operations.

Our solutions of Sections 3 and Section 3.2 only support a single merging operation. In Section 1.5 we discuss a construction supporting unbounded messages from new assumptions. Anyway, we stress that also a mFE supporting single merge is already sufficient for most of our applications like the implications of MI-FE, i$\mathcal{O}$, and all other applications when limited to a single operation. Furthermore, it is really trivial to generalize our constructions for single merging operation to support a *bounded* number q of merging operations with the drawback of having parameters growing as q. mFE schemes supporting only q merging operations are sufficient to build $(q-1)$-inputs MI-FE schemes and homomorphic obfuscators supporting $q-2$ merging operations.

Our security notion. We adopt as security definition an indistinguishability-based (IND) one. Recall that in the standard IND-Security game for FE an efficient adversary can output two challenge messages m_0 and m_1 and can ask any token for a function f that does not allow to distinguish the two messages, i.e., such that $f(m_0) = f(m_1)$. This is to avoid trivial attacks. In mFE this constraint is not sufficient. In fact, it could be that $f(m_0) = f(m_1)$ but $f(m||m_0) \neq f(m||m_1)$ allowing the adversary to distinguish by merging the challenge ciphertext with a ciphertext encrypting m. Notice that this situation is similar to the case of MI-FE. Therefore, we change the definition in the obvious way, by generalizing the above constraint to take in account any sequence of messages that "extends" the challenge messages (in poor words, to any message that has the challenge message as substring). More formally we also allow the challenge to be a pair of sequences (s_0, s_1) of merging operations with the same length and structure. Furthermore, we relax the security to the the *selective* model where the adversary has to declare its challenge at beginning of the game, i.e., before receiving the public-key. This is because, as standard in many works in the area, the selective model simplifies the security reductions. However, we remark that the selective model is sufficient to imply iO, homomorphic iO, riO, selectively IND-Secure MI-FE, and the kind of homomorphic CCA1-secure PKE described in Section 1.2. Details on our security notion are given in Section 2.2.

1.4 Overview of the construction and of the security reduction.

We assume that the reader is familiar with the construction of FE of Garg *et al.* [GGH$^+$13].

Our construction for single merging operation. Let us now sketch how to construct mFE scheme that supports a single merging operation. The technical details of our construction are tightly related with the need of proving its security so we will explain it step by step following an informal approach. Let us call ciphertexts of "first level" the ciphertexts encrypting directly messages and not resulting from merge operations. Suppose that in our scheme the ciphertexts of first level have the same form as in the Garg *et al.*'s scheme. Consider now the following implementation of the merge operation. To merge two ciphertexts (c_1, c_2, π_1) and (c_3, c_4, π_2), re-encrypt c_1 and c_3 under the first public-key to get Ct_1 and c_2 and c_4 under the second public-key to get Ct_2 adding a proof that Ct_1 and Ct_2 encrypt respectively strings $(c_1||c_3)$ and $(c_2||c_4)$ such that there exist proofs π_1, π_2 of the fact that c_1 and c_2 encrypt the same message and c_3 and c_4 encrypt the same message with respect to some proof systems to be specified later. We call the $\mathcal{NP}$ language consisting of statements of such form L^2 to distinguish it from L^1, the language used for the ciphertexts of first level (consisting of pairs of ciphertexts that encrypt the same message). That is, the new ciphertext will be $(\mathsf{Encrypt}(\mathsf{Pk}_1, (c_1, c_3)), \mathsf{Enc}(\mathsf{Pk}_2, (c_2, c_4), \pi)$, where π is a proof of the fact that Ct_1 and Ct_2 belong to L^2. To simplify the construction, let us use the following tools. First of all, we employ (1) a non-interactive witness indistinguishable proof systems (NIWI) with statistical soundness [FLS90] and (2) a statistically binding commitment scheme. In the public-key we put two CRSs of the NIWI system, one used to prove statements in L^1 and one for L^2, and a statistically binding commitment com. Actually for the needs of our reduction, the languages L^1 and L^2 are changed as follows. In the real scheme the NIWI is used to prove the following statement: "the pair of ciphertexts $(\mathsf{Ct}_1, \mathsf{Ct}_2)$ are well formed (as specified before) or com is a commitment to $(\mathsf{Ct}_1||\mathsf{Ct}_2)$". Let us now sketch the security reduction. Recall that the adversary $\mathcal{A}$ selects as challenges two sequences $(s_0 = (m_1, m_2), s_1 = (m_3, m_4))$ of merging operations and let us denote by x_b^1 (resp. x_b^2) the *inner* string encrypted in the ciphertext *induced* by the sequence s_b with respect to the first (resp. second) public-key. The definition is better explained by an example. For instance, with respect to the sequences s_0, x_0^1 is string $(\mathsf{Encrypt}(\mathsf{Pk}_1, m_1)||\mathsf{Encrypt}(\mathsf{Pk}_1, m_2))$. Furthermore, let use denote by $x_b^i[L]$ (resp. $x_b^i[R]$) the left part (resp. right part) of x_b^i, e.g., in the previous example $x_0^1[L] = \mathsf{Encrypt}(\mathsf{Pk}_1, m_1)$ and $x_0^1[R] = \mathsf{Encrypt}(\mathsf{Pk}_1, m_2)$. Note that the challenge ciphertext Ct consists of $(\mathsf{Ct}_1, \mathsf{Ct}_2, \pi)$ where, if for instance the challenge bit $b = 0$, Ct_1 encrypts x_0^1 and Ct_2 encrypts x_0^2 and π is a proof that $(\mathsf{Ct}_1, \mathsf{Ct}_2) \in L^2$ that is in turn relative to the two proofs π_1', π_2'. The latter proofs are such that (1) π_1' is a proof that $x_0^1[L]$ and $x_0^2[L]$ encrypt the same message (in the previous example m_1) and (2) π_2' is a proof that $x_0^1[R]$ and $x_0^2[R]$ encrypt the same message (in the previous example m_2).

Consider the following series of hybrid experiments. In the first experiment H_0, the sequence s_0 is chosen as challenge. In the security reduction we first switch to an hybrid H_1 where com is a commitment to $(\mathsf{Ct}_1||\mathsf{Ct}_2)$. Then, we switch to an hybrid H_2 where the NIWI proof is computed with respect to the randomness used for com (which, by definition, represents a valid witness). Then, in H_3, Ct_1 is encryption of x_0^1 as in the previous experiment but Ct_2 is changed to be encryption of x_1^2 where x_1^2 is the ciphertext induced by the sequence s_1 with respect to the second public-key. The indistinguishability of the last two experiments follows by the security of the PKE scheme. Then, in H_4, any token is changed to be an obfuscation of a machine that uses Sk_2 instead of Sk_1. The indistinguishability of H_4 from H_3 follows observing that, by the statistical soundness of the NIWI and by the correctness of the PKE scheme, the only ciphertexts that do not encrypt the same inner-message are associated with NIWI proofs produced using the "trapdoor" witness, i.e., the witness of the fact that com is a commitment to $(\mathsf{Ct}_1||\mathsf{Ct}_2)$. By the statistical binding property of the commitment scheme, by the correctness of the PKE scheme, by definition of L^2 and by statistical soundness of the NIWI system,, w.v.h.p. the *only* pair of ciphertexts that encrypts different inner-messages is the one in the challenge ciphertext, but the messages $M_0 = (m_1, m_2)$ resulting from decrypting Ct_1 recursively using Sk_1 and the message $M_1 = (m_3, m_4)$ resulting from decrypting Ct_2 recursively using Sk_2, are such that, by definition of the game, $f(M_0) = f(m_1||m_2) = f(m_3||m_4) = f(M_1)$ where f is the function associated with the token. Thus, we can invoke the security of iO to argue the indistinguishability of the last two hybrids. Then, in H_5, Ct_2 is encryption of x_1^2 as in the previous experiment but Ct_1 is changed to be encryption of x_1^1 where x_1^1 is the ciphertext induced by the sequence s_1 with respect to the first public-key. The indistinguishability of the last two experiments follows by the security of the PKE scheme. Then, in H_6, any token is changed to be an obfuscation of a machine that uses Sk_1 instead of Sk_2 (that is, it uses the original machine) and the indistinguishability of the last two hybrids is symmetrical to that of H_4 from H_3. Finally in H_7 and H_8 the witness used to compute the proof in the challenge ciphertext and the commitment com are turned back to the original ones. The latter experiment corresponds to the experiment in which s_1 is chosen as challenge as desired.

The construction can be also extended to support single merging operation but for messages of unbounded length as follows. First, note that in order to support multiple merging operation, we have to handle messages and thus ciphertexts of *unbounded* size. Therefore, we need (1) to assume obfuscation for TMs with unbounded inputs that was recently proposed by Ishai *et al.* [IPS14] and (2) assume that the obfuscator is public-coin differing-inputs secure as defined in the latter work. In fact, whereas in the previous security reduction we put in the public-key a commitment to the challenge ciphertext, we would, in the case of unbounded inputs, commit to the *hash* of such challenge with respect to a collision-resistant hash function. The security reduction is identical except that we have to take in account collisions to the committed value. Specifically, whereas in the bounded case the two obfuscated machines have same i/o, in the unbounded case instead the only inputs that would allow to distinguish the two machines are the collisions to the committed value. To that aim, we cshow that this family is public-coin differing-inputs secure (without any auxiliary input) based on the security of the hash function. It is easy to observe that the construction can be extended to support a bounded number of operations with ciphertexts that can possibly blow-up in size.

1.5 mFE for multiple merging operations

A natural approach to generalize our scheme to support multiple merging operation would be to simply iterate the re-encrypt approach (but also similar approaches we tempted suffer from the same problems) adding proofs of "well-formedness" relative to the *previous* proofs in a recursive way. Apart from technical problems arising from the need for proof systems capable to handle any $\mathcal{NP}$-statement, and thus for the universal relation [Mic00, BG08], the most serious hurdle is that we have no guarantee that in this case the size of the proofs does not grows by a multiplicative factor: for instance the size could double after any merging operation limiting the number of operations to logarithmic. A patch could be to resort to *succinct* arguments [Kil92, Mic00, Gro10], an approach followed in Boneh *et al.* [BSW12] (see also the related work of Chase *et al.* [CKLM13]), but as in the last work we would need to introduce special knowledge assumptions and moreover we would incur in the same limitation of having a bounded number of merging operations with either the size of the public-key being linear in the bound or both the size of public-key and ciphertexts being logarithmic in that bound. For such reasons, we did not further explore these directions and we propose a drastically different approach. Recall that one source of the problems arises from the fact that in order to use a proof system for $\mathcal{NP}$ multiple times, the merged ciphertext $\mathsf{Ct} = (\mathsf{Ct}_1, \mathsf{Ct}_2)$ should be such that Ct_1 and Ct_2 encrypt the same message so to be able to add a traditional proof of well-formedness. This clashes with the fact that Ct_1 and Ct_2 have to be encrypted with different public-keys. One solution could be to encrypt in Ct_2 the same message encrypted in Ct_1 but then in the security reduction we would have to switch to an obfuscated machine that incorporates *both* Sk_2 (to decrypt the outer-ciphertext) and Sk_1 (to decrypt the inner-ciphertexts recursively). In this case, the reduction would break down in showing the indistinguishability of H_5 from H_4, i.e., in switching the "left" ciphertext produced with the first public-key (because we have to simulate a machine that incorporates the first secret-key). Another possibility could be to adopt the punctured programming approach of Waters [Wat14] but this would incur in more problems, the most serious being that the Waters' techniques do not seem well suited to handle messages of variable length. We propose to draw upon new assumptions related to *average-case* obfuscation introduced by Hohenberger *et al.* [HRSV11]. In the latter work, it was proposed a notion of obfuscation aimed to model functionalities like the re-encryption one. To be useful in our context, we have to generalize their functionality to be *recursive* and add a signature. In fact, we need to re-encrypt under Pk_2 a nested encryption (for example $\mathsf{Encrypt}(\mathsf{Pk}_1, (\mathsf{Encrypt}(\mathsf{Pk}_1, m_1), \mathsf{Encrypt}(\mathsf{Pk}_1, m_2))))$ under Pk_1. This would be not still sufficient. In fact having a obfuscated re-encryption machine that translates ciphertexts from Pk_1 to Pk_2 it should be possible to show the indistinguishability of two ciphertexts encrypted under Pk_1, *but* in our case the obfuscated machine also needs Sk_2. Consider a more sophisticated functionality that (1) takes as input two pairs of ciphertexts (c_1, c_2) and (c_3, c_4) and (2) recursively decrypts c_1 and c_2 with Sk_1, (2) decrypts (with Sk_1) and re-encrypts the inner-messages contained in c_3 and c_4 under Pk_2 (using the same structure as c_1 and c_2) to get c_1', c_2' and (3) if c_1 has the same structure as c_3 and c_2 has the same structure as c_4 then outputs c_1', c_2' and a signature of $c_1||c_2||c_1'||c_2'$, otherwise output an error $\perp$. What means for two ciphertexts C_1 and C_2 to have the same structure is better explained by the following examples. For instance, suppose that C_1 is an encryption of (d_1, d_2) where d_2 is an encryption of m_1 and d_1 is an encryption of (e_1, e_2) with e_1 and e_2 encrypting respectively m_2 and m_3, and where all ciphertexts are encrypted under Pk_1. In addition, suppose that C_2 is an encryption of (d'_1, d'_2) where d'_2 is an encryption of m_1 and d'_1 is an encryption of (e'_1, e'_2) with e'_1 and e'_2 encrypting respectively m_2 and m_3, and where all ciphertexts are encrypted under Pk_2. Then C_1 and C_2 have the same structure. Suppose instead that C_1 is as before but C_2 is an encryption of (d'_1, d'_2) where d'_2 is an encryption of $m'_1 \neq m_1$ and d'_1 is an encryption of (e'_1, e'_2) where e'_1 and e'_2 encrypt respectively m_2 and m_3, where all ciphertexts are encrypted under Pk_2. Then, C_1 and C_2 have different structures. Intuitively the functionality checks that the ciphertexts have the same structure to "authenticate" only the ciphertexts encrypting (recursively) the same sequence of messages, and thus acting as a proof of well-formedness. We call this functionality "recursively decrypt and re-sign-encrypt" (RDRSE). As first step, let us put forward the assumption that the obfuscator for TMs of Ishai *et al.* based on the obfuscator of Garg *et al.* is an average-case obfuscator for the class of machines that implement RDRSE. Then we change our previous mFE scheme as follows. Let $M[\mathsf{Sk}_1, \mathsf{Pk}_2, \mathsf{sk}](\cdot, \cdot)$ be a average-case obfuscation of a RDRSE machine that uses a signature scheme with signing key sk incorporated in it. To merge two ciphertexts (c_1, c_2, π_1) and (c_3, c_4, π_2), re-encrypt c_1 and c_3 under the first public-key to get Ct_1, computes $M[\mathsf{Sk}_1, \mathsf{Pk}_2, \mathsf{sk}]((\mathsf{c}_1, \mathsf{c}_3), (\mathsf{c}_2, \mathsf{c}_4))$ to get (c'_2, c'_4) and a signature σ and encrypt (c_1, c_3) under the first publick-key to get Ct_1 and (c'_2, c'_4) under the second public-key to get Ct_2 and add a proof of the fact that Ct_1 and Ct_2 are such that either (1) Ct_1 encrypts (c_1, c_2) and Ct_2 encrypts c'_2, c'_4 and a signature σ of $c_1||c_2||c'_2||c'_4$ or (2) the hash of $(\mathsf{Ct}_1, \mathsf{Ct}_2)$ is committed in com. Furthermore, we now require that the machines corresponding to the tokens are obfuscated with a diO secure against adversaries with auxiliary input (note that now the tokens are changed in the obvious way and incorporate the verification key corresponding to sk). The security reduction follows the previous lines except (1) for the indistinguishability of the experiments in which we switch the secret keys in the tokens and (2) for the indistinguishability of the experiments in which we switch the ciphertexts. In the case (1) we would like to prove that for the two machines (one using Sk_1 and one using Sk_2) it is difficult to find a distinguishing input. Intuitively this holds since the RDSME machine is obfuscated with an average-case obfuscator and thus should act as a black-box. In the second case we would like to argue that an IND-CPA adversary having in addition an average-case obfuscation of an RDRSE machine has still negligible advantage in distinguishing the challenge ciphertexts. This should hold for the same reason. Though these informal arguments seem reasonable, we were not able to reduce the security only to our previous assumption combined with known assumptions and thus we restrict to conjecture that the sketched construction is secure, defering to future works or versions of this paper a formal treatment. We conclude with a last remark. Even if it was possible to prove the security of the above sketched construction, the underlying PKE scheme should have the additional property that a ciphertext encrypting a message of length n has length $O(n + \mathsf{poly}(\lambda)$, where λ is the security parameter, and not for instance $O(n \cdot \lambda)$. In fact, in the latter case, encrypting the ciphertexts recursively we would incur in an exponential blow-up. It is easy to construct such a PKE scheme: for example setting the encryption of m to $\mathsf{Encrypt}(\mathsf{Pk}, r), \mathsf{PRG}^{|m|}(r) \oplus m)$ where $\mathsf{Encrypt}$ is an arbitrary PKE scheme that encrypts string of fixed size and $\{\mathsf{PRG}^n\}_n$ is a family of pseudo-random generators that expands λ bits to n bits.

1.6 Related work.

The primitive we introduce is novel yet it is related to the following cryptographic objects.

- mFE shares with homomorphic encryption (see [Gen09]) the capability of "combining" different ciphertexts in an oblivious way but it extends homomorphic encryption with functional features. As discussed before, in some sense mFE achieves the best of possible homomorphism capabilities for FE.

- mFE shares with MI-FE [GGG+14] the capability of computing over *multiple* encrypted data and indeed mFE enables all the applications of MI-FE but the converse is not true. In particular, MI-FE differs from mFE in an essential aspect. When two ciphertexts c_1 and c_2 are merged with mFE, it is not longer possible to recover the original information. Instead, while MI-FE allows to compute a function f over two ciphertexts c_1 and c_2 yet it is possible to swap c_2 with any other ciphertext c_3 to compute f over c_1 and c_3. Furthermore, mFE supporting unbounded number of merging operation implies MI-FE for functions of *unbounded arity*. Instead, in known MI-FE schemes the arity of the supported functions is fixed at setup time. We also remark that it is not known how to use MI-FE to re-encrypt so to imply mFE.

- FE for randomized functionalities was studied by Goyal *et al.* [GJKS13] who presented two types of notions, one simulation-based and one indistinguishability-based. The simulation-based one can be defined only in limited settings due to known impossibility results [BSW11, AGVW13], and moreover, to be useful in constructing mFE, simulation-based secure FE for randomized functionalities should be generalized to 2-inputs for which more severe impossibility results are known as it would imply virtual black-box obfuscation [BGI+01]. Instead, the indistinguishability-based notion is limited to statistically indistinguishable distributions, thus excluding the re-encryption functionality.

- Target malleability introduced by Boneh *et al.* [BSW12] restricts the set of homomorphic operations one can perform on encrypted data but does *not* offer functional features. Nevertheless, the techniques used in their construction could be useful for mFE, albeit to avoid their strong assumptions and limitation on the size of the public-key and ciphertexts we depart from their approach.

- As discussed in Section 1.2, mFE implies CCA1-secure PKE with mergeable properties. Loft *et al.* [LMSV12] present a construction of somewhat homomorphic CCA1-secure PKE from special knowledge assumptions. Prabhakaran and Rosulek [PR08] also present alternative notions of CCA-security for homomorphic encryption.

- Gentry *et al.* [GSW13] construct an attribute-based encryption (ABE) scheme with fully homomorphic properties. In ABE the ciphertext is associated with a pair (m, x) where x is public and only m is hidden. The system of Gentry *et al.* allows homomorphic operations only on m, and thus is incomparable to ours.

2 Definitions

A *negligible* function $\mathsf{negl}(k)$ is a function that is smaller than the inverse of any polynomial in k. If D is a probability distribution, the writing "$x \leftarrow D$" means that x is chosen according to D. If D is a finite set, the writing "$x \leftarrow D$" means that x is chosen according to uniform probability on D. If $q > 0$ is an integer then $[q]$ denotes the set $\{1, \ldots, q\}$. All algorithms, unless explicitly noted, are probabilistic polynomial time and all adversaries are modeled by non-uniform polynomial time algorithms. If B is an algorithm and A is an algorithm with access to an oracle then A^B denotes the execution of A with oracle access to B. If a and b are arbitrary strings, then $a||b$ denotes the string representing their delimited concatenation. If M

is a Turing machine, we denote by $\mathsf{steps}(M, x)$ the number of steps executed by M on input x before halting or $\perp$ if it never halts.

2.1 Functional Encryption

Functional encryption schemes are encryption schemes for which the owner of the master secret can compute restricted keys, called *tokens*, that allow to compute a *functionality* on the plaintext associated with a ciphertext. We start by defining the notion of a functionality.

Definition 2.1 [Functionality] A *functionality* F is a function $F : K \times M \to \Sigma$ where K is the *key space*, M is the *message space* and Σ is the *output space.*

In this work, our FE schemes are for the following functionality.

Definition 2.2 [TM Functionality] The TM functionality has key space K equals to the set of all Turing machines which accept any input of unbounded polynomial length and halt in polynomial time. The message space M is the set $\{0, 1\}^*$. For $M \in K$ and $m \in M$, we have $\mathsf{TM}(M, m) = (M(m), t)$, where t is the number of time steps that M performs on input m.

Remark 2.3 In case of a scheme with input-specific run time (cf. Definition E.1), we also require that the functionality outputs the run time of machine M on m along with the output of the computation $M(m)$.

The definition of a FE scheme can be found in Appendix E along with the standard definition of indistinguishability-based security for it.

2.2 Mergeable Functional Encryption

In this Section we will formally introduce the concept of mergeable functional encryption (mFE).

Our notation for merging operations. We first introduce our notation related to merging operations. To not overburden the discussion, we prefer to adopt a non too much formal style but expanding the definitions with concrete examples.

We denote the merge of string x_1 with string x_2 by (x_1, x_2)[2] and we call such string (x_1, x_2) a *sequence.* (We assume implicitly that the symbols '(',')' and ',' do not belong to the string alphabet and thus sequences can be parsed efficiently. This can be formalized in standard ways but we over skip these details). Furthermore, we define a sequence of merging inductively in the obvious way. That is, a string in the message space is a sequence of merging operations and if s_1 and s_2 are sequences of merging operations, then (s_1, s_2) is a sequence of merging operations. For instance, the sequence $s = ((x_1, (x_2, x_3)), x_4)$ is the result of (1) merging x_2 with x_3 to get sequence s_1, then (2) merging x_1 with s_1 to get sequence s_2 and finally (3) s_2 with x_4 to get sequence s. Note that in this paper with a slight abuse of notation sometimes we use the notation $(x.y)$ and similar to denote both sequences of merging operations and lists of elements. We say that a sequence of merging operations s *splits over* the strings $(x_1, \ldots, x_n)$ if such variables appear in s in that order. For instance $(x_1, (x_2, (x_3, x_4)))$ splits over (x_1, x_2, x_3, x_4). We say that a sequence of merging operations s *ranges over* the set of strings M if the strings appearing in s belong to M. For instance $(x_1, (x_2, (x_3, x_2)))$ ranges over *any* set containing $\{x_1, x_2, x_3\}$. If s is a sequence we denote by $\mathsf{cat}(s)$ the concatenation of the strings which the sequence consists of. For instance if $s = ((x_1, (x_2, x_3)), x_4)$ then $\mathsf{cat}(s) = x_1||x_2||x_3||x_4$.

[2]Notice that the order is important, so the operation (x_1, x_2) is different from (x_2, x_1).

We say that a sequence of merging operations s_1 splitting over the strings $(x_1, \ldots, x_n)$ has the same *structure* of a sequence of merging operations s_2 splitting over $(y_1, \ldots, y_m)$ if (1) $m = n$ and (2) if s_1' is the string obtained by replacing any string x_i in s_1 with $0^{|x_i|}$ and s_2' is the string obtained by replacing any string y_i in s_2 with $0^{|y_i|}$, then $s_1' = s_2'$. Note that this implies that for any i, $|x_i| = |y_i|$. For instance, $(010,(111,001))$ does *not* have the same structure of $((010,111),001)$, $(010,(111,001))$ does *not* have the same structure of $(010,(11,001))$, but $(010,(11,001))$ *does* have the same structure of $(111,(00,110))$. That is, two sequences have the same structure if the sequence of parenthesis and commas correspond and any string has the same length. We say that $t(\cdot)$ is a function of merging operations if $t(\cdot)$ is a sequence of merging operations containing at least one special symbol $\star$ and if s is a sequence of merging operations then $t(s)$ is the result of replacing $\star$ with s in t. Furthermore, If $t(\cdot) = \star$ then $t(s) = s$ for any sequence of merging operations s. For instance, if $t(\cdot) = ((x_1,(x_2,*)),x_3)$ and $s = (x_3, x_4)$ then $t(s) = ((x_1,(x_2,(x_3,x_4))),x_3)$. We say that a function of merging operations $t(\cdot)$ *ranges over* the set of strings M if except the symbol $\star$ the strings appearing in t belong to M. For instance $t(\cdot) = (x_1,(\star,(x_3,x_2)))$ ranges over *any* set containing $\{x_1, x_2, x_3\}$. Finally, we stress that the strings in the sequence of merging operations we will consider will be both messages in the mmessage space and ciphertexts. For instance, if c_1, c_2, c_3 and c_4 are ciphertexts, then $s = ((c_1,c_2),(c_3,c_4))$ is a sequence of merging operations on ciphertexts.

Definition 2.4 [mergeable Functional Encryption Scheme] A *mergeable functional encryption* scheme mFE for functionality F is a tuple $\mathsf{mFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Merge})$ of 5 algorithms with the following syntax:

1. $\mathsf{Setup}(1^\lambda)$ outputs *public* and *master secret* keys $(\mathsf{Pk}, \mathsf{Msk})$ for *security parameter* λ.

2. $\mathsf{KeyGen}(\mathsf{Msk}, k)$, on input a master secret key Msk and *key* $k \in K$ outputs *token* $\mathtt{Tok}$.

3. $\mathsf{Enc}(\mathsf{Pk}, m)$, on input public key Pk and *plaintext* $m \in M$ outputs *ciphertext* Ct: Furthermore, if s is a sequence of merging operations splitting over the strings $(x_1, \ldots, x_n)$, with a slight abuse of notation, we denote by $\mathsf{Enc}(\mathsf{Pk}, s)$, the sequence s' that is equal to s except that for any $i \in [n]$, any occurrence of x_i is replaced by $\mathsf{Enc}(\mathsf{Pk}, x_i)$. For instance, $\mathsf{Enc}(\mathsf{Pk},(x_1,(x_2,x_3))) = (c_1,(c_2,c_3))$ where c_1, c_2 and c_3 are respectively the encryptions of x_1, x_2 and x_3. (We stress that even if a sequence s contains two equal strings, e.g., $(x_1,(x_2,x_1))$, in the encryption of s any ciphertext is generated with *independent* randomness).

4. $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok})$ outputs $y \in \Sigma \cup \{\perp\}$.

5. $\mathsf{Merge}(1^\lambda, \mathsf{Pk}, \mathsf{Ct}_1, \mathsf{Ct}_2)$: if Ct_1 and Ct_2 are ciphertexts, the algorithm outputs a ciphertext Ct resulting from merging Ct_1 with Ct_2.

   With a slight abuse of notation, we also define Merge to take as input a sequence of merging operations on ciphertexts in the obvious way, described by the following example. Let $s_1 = (c_1,(c_2,c_3))$ be a sequence of merging operations on ciphertexts, then

   $\mathsf{Merge}(1^\lambda, \mathsf{Pk}, s) \overset{\triangle}{=} \mathsf{Merge}(1^\lambda, \mathsf{Pk}, c_1, \mathsf{Merge}(1^\lambda, \mathsf{Pk}, c_2, c_3))$.

In addition we make the following requirements:

- *Correctness*: For all $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, all $k \in K$ and any sequence s splitting over strings $\in M$ such that $\mathsf{cat}(s) \in M$, for $\mathtt{Tok} \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k)$ and $\mathsf{Ct} \leftarrow \mathsf{Merge}(1^\lambda, \mathsf{Pk}, \mathsf{Enc}(\mathsf{Pk}, s))$, we have that $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok}) = F(k, \mathsf{cat}(s))$ whenever $F(k, \mathsf{cat}(s)) \neq \perp$, except with negligible probability in λ. (See [BO13] for a discussion about this condition.)

- *Compactness*: there exists a polynomial poly such that for all $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, for any sequences s_1, s_2 splitting over strings $\in M$ such that $\mathsf{cat}(s_1), \mathsf{cat}(s_2) \in M$, for $\mathsf{Ct}_1 \leftarrow \mathsf{Merge}(1^\lambda, \mathsf{Pk}, \mathsf{Enc}(\mathsf{Pk}, s_1))$, $\mathsf{Ct}_2 \leftarrow \mathsf{Merge}(1^\lambda, \mathsf{Pk}, \mathsf{Enc}(\mathsf{Pk}, s_2))$, $\mathsf{Ct} \leftarrow \mathsf{Merge}(1^\lambda, \mathsf{Pk}, \mathsf{Ct}_1, \mathsf{Ct}_2)$, we have that $|\mathsf{Ct}| \in O(|\mathsf{Ct}_1| + |\mathsf{Ct}_2| + \mathsf{poly}(\lambda))$. Note that here sequences s_1, s_2 might be a single message.

Remark 2.5 Notice that the correctness requirement has as special case the correctness for traditional FE.

We consider also the following two properties for mFE schemes for Turing machines.

- *Succinctness:* A mFE scheme for TM is said to be succinct if the ciphertexts have size polynomial in the security parameter and in the message size, and the tokens generated using KeyGen for machine M have size $q(\lambda, |M|)$, where q is a polynomial and $|M|$ is the size of the Turing machine.

- *Input-specific running-time:* A mFE scheme for TM is said to have input-specific run time if the decryption algorithm on input token $\mathtt{Tok}$ for machine M and encryption of message m, takes time $p(\lambda, \mathsf{steps}(M, m))$, where p is a polynomial.

Indistinguishability-based security. The indistinguishability-based notion of security for mergeable functional encryption scheme
$\mathsf{mFE} = (\mathsf{mFE.Setup}, \mathsf{mFE.KeyGen}, \mathsf{mFE.Enc}, \mathsf{mFE.Eval})$ for functionality F defined over (K, M) is formalized by means of the following game parametrized by a polynomial poly:

$\mathsf{mIND}_{\mathcal{A}}^{\mathsf{mFE},\mathsf{poly}}$ between an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and a *challenger* $\mathcal{C}$. Below, we present the definition for only one message; it is easy to see the definition extends naturally for multiple messages.

$\mathsf{mIND}_{\mathcal{A}}^{\mathsf{mFE,poly}}(1^\lambda)$

1. $\mathcal{C}$ generates $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{mFE.Setup}(1^\lambda)$ and runs $\mathcal{A}_0$ on input Pk;
2. $\mathcal{A}_0$ submits queries for keys $k_i \in K$ for $i = 1, \ldots, q_1$ and, for each such query, $\mathcal{C}$ computes $\mathtt{Tok}_i = \mathsf{mFE.KeyGen}(\mathsf{Msk}, k_i)$ and sends it to $\mathcal{A}_0$.

   When $\mathcal{A}_0$ stops, it outputs two *challenge sequences of merging operations* s_0, s_1 and its internal state $\mathtt{st}$.
3. $\mathcal{C}$ picks $b \in \{0,1\}$ at random, computes the *challenge ciphertext* $\mathsf{Ct} = \mathsf{mFE.Merge}(\mathsf{Pk}, \mathsf{mFE.Encrypt}(\mathsf{Pk}, s_b))$[a] and sends Ct to $\mathcal{A}_1$ that resumes its computation from state $\mathtt{st}$.
4. $\mathcal{A}_1$ submits queries for keys $k_i \in K$ for $i = q_1 + 1, \ldots, q$ and, for each such query, $\mathcal{C}$ computes $\mathtt{Tok}_i = \mathsf{mFE.KeyGen}(\mathsf{Msk}, k_i)$ and sends it to $\mathcal{A}_1$.
5. When $\mathcal{A}_1$ stops, it outputs b'.
6. **Output:** The challenger outputs 1 (i.e., the adversary wins the game) iff the following conditions are all satisfied:
   (a) $b = b'$.
   (b) s_0 and s_1 are sequences of strings over strings in $\mathcal{M}$ satisfying $|s_0| = |s_1|$.
   (c) for any function of merging operations $t(\cdot)$ of length $\leq \mathsf{poly}(\lambda)$ ranging over $\mathcal{M}$ it holds that: $F(k_i, t(s_0)) = F(k_i, t(s_1))$ for $i = 1 \ldots, q$.

[a]We refer the reader to the definition of the procedure Merge for the use of this notation.

For any polynomial poly, the advantage of adversary $\mathcal{A}$ in the above game parametrized by poly is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{mFE,mIND,poly}}(1^\lambda) = |\mathrm{Prob}[\mathsf{mIND}_{\mathcal{A}}^{\mathsf{mFE,poly}}(1^\lambda) = 1] - 1/2|.$$

Definition 2.6 We say that mFE is *indistinguishably secure* (IND-Secure, for short) for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a polynomial poly such that the quantity $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{mFE,mIND,poly}}(1^\lambda)$ is negligible in λ.

Definition 2.7 [Selective IND-Security] The selective security game of mFE is similar to the above game except that the adversary has to declare the challenges at the beginning of the experiment. We say that a mFE scheme is selectively IND-Secure if any PPT adversaries has at most negligible advantage in such game.

On selective security. We show the security of our constructions in the selective IND-Security model. This is because, as standard in many works in the area, the selective model simplifies the security reductions. However, we remark that the selective model is sufficient to imply iO, homomorphic iO, riO, selectively IND-Secure MI-FE, and the kind of homomorphic CCA1-secure PKE described in Section 1.2.

On simulation-based security. A stronger notion of security for FE is the simulation-based one [BSW11, O'N10] that fixes the deficiency of the indistinguishability-based one. In Appendix

E we recall the simulation-based security (SIM-Security) for FE. Unfortunately, several impossibility results for SIM-Security for FE are known [AGVW13, BSW11, DIJ$^+$13, CI13, HW15]. Notwithstanding, Decaro *et al.* [DIJ$^+$13] show how to obtain some limited forms of SIM-Security for FE. We remark that even weak forms of SIM-Security for mFE are unattainable as SIM-Secure mFE implies VBB obfuscation [BGI$^+$01], for the same reason that SIM-Secure MI-FE [GGG$^+$14] is impossible. Furthermore, we stress that the recent transformation for SIM-Secure FE in the Random Oracle model of Iovino and Żebrowski [IZ14] completely breaks down in the case of mFE as well as for 2-inputs FE.

mFE for one merging operation. We provided definitions for mFE supporting unbounded number of operations. It is straight-forward to adapt them to the case of mFE supporting one merging operation as needed in Section 3. Moreover, for mFE supporting one merging operation the compactness property is not required.

2.3 Building blocks

In Appendix C and D we recall the definitions of indistinguishability obfuscation (iO) and public-coin differing-inputs obfuscation (diO). In Appendix B we recall the notion of NIWI proof systems.

3 Our mFE scheme for one merging operation

Definition 3.1 [MFE scheme for one merging operation] Let $\mathsf{NIWI}^i = (\mathsf{CRSGen}^i, \mathsf{Prove}^i, \mathsf{Verify}^\mathsf{i})$ for $i = 1, 2$ be two NIWI proof systems (cf. Section B) for some $\mathcal{NP}$-languages to be specified later, Com a (perfectly binding) commitment scheme (cf. Appendix A.3), $\mathsf{E} = (\mathsf{E.Setup}, \mathsf{E.Encrypt}, \mathsf{E.Decrypt})$ a PKE scheme (cf. Appendix A.2), and $\mathsf{i}\mathcal{O}$ (cf. Section C) an iO for TMs with bounded inputs. Let $n(\lambda)$ be a bound on the size of the messages that our mFE has to support and $m(\lambda)$ a bound on the size of ciphertexts of E that encrypt messages of length $n(\lambda)$. We define a mFE scheme $\mathsf{mFE}[\mathsf{NIWI}^1, \mathsf{NIWI}^2, \mathsf{Com}, \mathsf{E}] = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Merge}, \mathsf{Eval})$ for functionality TM as follows.

- $\mathsf{Setup}(1^\lambda)$[3]: runs $(\mathsf{Pk}_1, \mathsf{Sk}_1) \leftarrow \mathsf{E.Setup}(1^\lambda)$, $(\mathsf{Pk}_2, \mathsf{Sk}_2) \leftarrow \mathsf{E.Setup}(1^\lambda)$, and $\mathsf{com} \leftarrow \mathsf{Com}(0^{2m(\lambda)})$, and sets $\mathsf{crs}^i \leftarrow \mathsf{CRSGen}^i(1^\lambda)^i$ for $i = 1, 2$. The procedure returns a pair $(\mathsf{Mpk}, \mathsf{Msk})$ where $\mathsf{Mpk} = (\mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{com}, \mathsf{crs}^1, \mathsf{crs}^2)$ and $\mathsf{Msk} = \mathsf{Sk}_1$.

- $\mathsf{Enc}(\mathsf{Pk}, m)$: on input $\mathsf{Pk} = (\mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{com}, \mathsf{crs}^1, \mathsf{crs}^2)$ and $m \in n(\lambda)$, the algorithm chooses randomness r_1 and r_2, and computes $\mathsf{Ct}_1 = \mathsf{E.Encrypt}(\mathsf{Pk}_1, m; r_1), \mathsf{Ct}_2 = \mathsf{E.Encrypt}(\mathsf{Pk}_2, m; r_2)$. Consider the following $\mathcal{NP}$-language[4]:

  $L^1 = \{(\mathsf{Ct}_1, \mathsf{Ct}_2) \in \{0,1\}^{2m(\lambda)} : \exists m, r, r_1, r_2 : (\mathsf{Ct}_1 = \mathsf{E.Encrypt}(\mathsf{Pk}_1, m; r_1) \ \wedge \ \mathsf{Ct}_2 = \mathsf{E.Encrypt}(\mathsf{Pk}_2, m; r_2)) \ \vee \ \mathsf{com} = \mathsf{Com}(\mathsf{Ct}_1 || \mathsf{Ct}_2; r)\}$.

  (Note that L^1 is relative to $\mathsf{com}, \mathsf{Pk}_1, \mathsf{Pk}_2$). The procedure outputs a ciphertext of first level $(1, \mathsf{Ct}_1, \mathsf{Ct}_2, \pi)$ where π is a proof of the fact that $(\mathsf{Ct}_1, \mathsf{Ct}_2) \in L^1$ computed with Prove^1 and crs^1 using as witness m, r_1, r_2.

- $\mathsf{KeyGen}(\mathsf{Msk}, M)$: on input Msk and a machine M, the algorithm outputs an iO of the following machine $T[M, \mathsf{Sk}_1, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{crs}^1, \mathsf{crs}^2, \mathsf{com}]$ as token.

[3]Formally, the procedure should also take as input the bound $m(\lambda)$ on the size of the messages (since it is used to generate the commitment) but for simplicity we omit such details.

[4]Formally we should define it as a family of languages indexed by the security parameter but henceforth for simplicity we omit this detail.

```
**Machine** $T[M, \mathsf{Sk}_1, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{crs}^1, \mathsf{crs}^2, \mathsf{com}](l, \mathsf{Ct}_1, \mathsf{Ct}_2, \pi)$
1. Pad with machine $T_2[M, \mathsf{Sk}_2, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{crs}^1, \mathsf{crs}^2, , \mathsf{com}]$
2. if $l = 1$ then do
3.     If $\mathsf{Verify}^1(\mathsf{crs}^1, (\mathsf{Ct}_1, \mathsf{Ct}_2), \pi) = 0$ then return $\perp$
4.     set $m = \mathsf{E.Decrypt}(\mathsf{Sk}_1, \mathsf{Ct}_1)$ and return $M(m)$
5. if $l = 2$ then do
6.     If $\mathsf{Verify}^2(\mathsf{crs}^2, (\mathsf{Ct}_1, \mathsf{Ct}_2), \pi) = 0$ then return $\perp$
7.     $(c_1, c_2) = \mathsf{E.Decrypt}(\mathsf{Sk}_1, \mathsf{Ct}_1)$
8.     set $m_1 = \mathsf{E.Decrypt}(\mathsf{Sk}_1, c_1)$ and $m_2 = \mathsf{E.Decrypt}(\mathsf{Sk}_1, c_2)$
9.     return $M(m_1||m_2)$
```

- $\mathsf{Merge}(\mathsf{Pk}, \mathsf{Ct}_1, \mathsf{Ct}_2)$: Let $\mathsf{Ct}_1 = (c_1, c_2, \pi_1)$ and $\mathsf{Ct}_2 = (c_3, c_4, \pi_2)$. The procedure sets $\mathsf{Ct}'_1 = \mathsf{E.Encrypt}(PK_1, c_1||c_3)$ and $\mathsf{Ct}'_2 = \mathsf{E.Encrypt}(PK_2, c_2||c_4)$. Consider the following $\mathcal{NP}$-language L^2:

  $L^2 = \{(\mathsf{Ct}_1, \mathsf{Ct}_2) \in \{0,1\}^{2m(\lambda)} : \exists c_1, c_2, c_3, c_4, \pi_1, \pi_2, r, r_1, r_2 : (\mathsf{Ct}'_1 = \mathsf{E.Encrypt}(\mathsf{Pk}_1, c_1||c_3; r_1) \wedge \mathsf{Ct}_2 = \mathsf{E.Encrypt}(\mathsf{Pk}_2, c_2||c_4; r_2) \wedge \mathsf{Verify}^1(\mathsf{crs}^1, \mathsf{c}_1||\mathsf{c}_2) = 1 \wedge \mathsf{Verify}^1(\mathsf{crs}^1, \mathsf{c}_3||\mathsf{c}_4) = 1) \vee \mathsf{com} = \mathsf{Com}(\mathsf{Ct}_1||\mathsf{Ct}_2; \mathsf{r})\}$.

  (Note that L^2 is relative to $\mathsf{com}, \mathsf{crs}^1, \mathsf{Pk}_1, \mathsf{Pk}_2$). The procedure computes a proof π of the fact that $(\mathsf{Ct}'_1, \mathsf{Ct}'_2) \in L^2$ using Prove^2 with crs^2 and witness $r_1, r_2, c_1, c_2, c_3, c_4, \pi_1, \pi_2$. The procedure outputs $(\mathsf{Ct}'_1, \mathsf{Ct}'_2, \pi)$.

- $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok})$: on input $\mathsf{Pk} = (\mathsf{Pk}_1, \mathsf{Pk}_2)$, $\mathsf{Ct} = (i, C_1, C_2, \pi)$ and $\mathtt{Tok} = \mathsf{i}\mathcal{O}(T[M, \mathsf{Sk}_1, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{crs}, \mathsf{com}])$, returns the output $\mathsf{Tok}(\mathsf{Ct})$ (i.e., evaluates the obfuscated program on input Ct).

It is straight-forward to see that the scheme is correct (and recall that in the context of one merging operation the compactness is not required).

3.1 Security reduction.

We assume that the reader is familiar with the overview presented in Section 1.4. We reduce the security of our mFE scheme to that of the underlying primitives via a series of hybrid experiments against a PPT adversary $\mathcal{A}$ attacking the selective IND-Security of mFE.

For simplicity we assume that the challenge sequences do *not* consist of single messages so that the challenge ciphertexts can not be of first level. However, it is easy to observe that in the latter case the reduction can be derived as special case of ours, but not to overburden the presentation we omit the details. Recall that the adversary $\mathcal{A}$ selects as challenges two sequences (s_0, s_1) of merging operations. We denote by x_b^1 (resp. x_b^2) the *inner* string encrypted in the ciphertext *induced* by the sequence s_b with respect to the first (resp. second) public-key. The definition is better explained by an example. For instance, if $s_0 = (m_1, m_2)$, then x_0^1 is the string $\mathsf{Encrypt}(\mathsf{Pk}_1, m_1)||\mathsf{Encrypt}(\mathsf{Pk}_1, m_2)$. Furthermore, we denote by $x_b^i[L]$ (resp. $x_b^i[R]$) the left part (resp. right part) of x_b^i, e.g., in the previous example $x_0^1[L] = \mathsf{Encrypt}(\mathsf{Pk}_1, m_1)$ and $x_0^1[R] = \mathsf{Encrypt}(\mathsf{Pk}_1, m_2)$. Note that in our construction the challenge ciphertext Ct with respect to the challenges (s_0, s_1) consist of $(\mathsf{Ct}_1, \mathsf{Ct}_2, \pi)$ where Ct_1 encrypts x_b^1 and Ct_2 encrypts x_b^2 and π is a proof that $(\mathsf{Ct}_1, \mathsf{Ct}_2) \in L^2$ that is in turn relative to (1) a NIWI proof π'_1 of the fact that $x_b^1[L]$ and $x_b^2[L]$ encrypt the same message and (2) a proof π'_2 of the fact that $x_b^1[R]$ and $x_b^2[R]$ encrypt the same message.

- H_0. This corresponds to the IND-Security game in which the chosen challenge sequence is s_0. Thus, the challenge ciphertext consists of $(\mathsf{Ct}_1, \mathsf{Ct}_2, \pi)$ where $(\mathsf{Ct}_1, \mathsf{Ct}_2)$ belong to L^2.

- H_1. This experiment is identical to H_0 except that the commitment com in the public-key is a commitment to $(\mathsf{Ct}_1, \mathsf{Ct}_2)$. Specifically, $\mathcal{A}$ selects its challenges $(s_0 = (m_1, m_2), s_1 = (m_3, m_4))$ and $x_0^1 = x_0^1[L]||x_0^1[R]$ and $x_0^2 = x_0^2[L]||x_0^2[R]$ are computed as described above along with (1) the NIWI proof π'_1 of the fact that $x_0^1[L]$ and $x_0^2[L]$ encrypt the same message (specifically m_1) and (2) the NIWI proof π'_2 of the fact that $x_0^1[R]$ and $x_0^2[R]$ encrypt the same message (specifically m_2). Then, the procedure Setup of mFE is run as in its definition except that com is set to be $\mathsf{com} = \mathsf{Com}((\mathsf{Ct}_1||\mathsf{Ct}_2); r)$ for some fresh randomness r where Ct_1 is an encryption of x_0^1 and Ct_2 is an encryption of x_0^2. The rest of the experiment can be simulated by means of Msk and Pk generated by the procedure Setup. Indeed, note that in the challenge ciphertext $\mathsf{Ct} = (\mathsf{Ct}_1, \mathsf{Ct}_2, \pi)$ the proof π is computed with respect to the proofs of "first level" π'_1, π'_2 and the randomness to encrypt x_0^1 in Ct_1 and x_0^2 in Ct_2.

  Claim 3.2 Indistinguishability of H_1 from H_0. It is easy to see that the claim follows from the computational hiding property of Com.

- H_2. This experiment is identical to H_1 except that the NIWI proof π in the challenge ciphertext $\mathsf{Ct} = (\mathsf{Ct}_1||\mathsf{Ct}_2)$ is computed with respect to the randomness r used to generate com.

  Claim 3.3 Indistinguishability of H_2 from H_1. It is easy to see that the claim follows from the witness indistinguishability property of NIWI observing that both the witness used in H_1 and the witness used in H_1 are valid witnesses of the fact that $(\mathsf{Ct}_1, \mathsf{Ct}_2) \in L^2$.

- H_3. This experiment is identical to H_2 except that Ct_2 is set to an encryption of x_1^2. The commitment com is still generated as $\mathsf{com} = \mathsf{Com}((\mathsf{Ct}_1||\mathsf{Ct}_2); r)$ and the randomness r is still used as witness to compute the proof π in the challenge ciphertext $\mathsf{Ct} = (\mathsf{Ct}_1, \mathsf{Ct}_2, \pi)$.

  Claim 3.4 Indistinguishability of H_3 from H_2. It is easy to see that the claim follows from the IND-CPA security of E observing that Sk_2 is *never* needed to simulate the experiment.

- H_4. This experiment is identical to H_3 except that any token is changed to be the obfuscation of the following machine:

```
**Machine** $T_2[M, \mathsf{Sk}_2, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{crs}^1, \mathsf{crs}^2, \mathsf{com}](l, \mathsf{Ct}_1, \mathsf{Ct}_2, \pi)$
1. Pad with machine $T[M, \mathsf{Sk}_1, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{crs}^1, \mathsf{crs}^2, \mathsf{com}]$
2. if $l = 1$ then do
3.     If $\mathsf{Verify}^1(\mathsf{crs}^1, (\mathsf{Ct}_1, \mathsf{Ct}_2), \pi) = 0$ then return $\perp$
4.     set $m = \mathsf{E.Decrypt}(\mathsf{Sk}_2, \mathsf{Ct}_2)$ and return $M(m)$
5. if $l = 2$ then do
```

```
6.    If $\mathsf{Verify}^2(\mathsf{crs}^2, (\mathsf{Ct}_1, \mathsf{Ct}_2), \pi) = 0$ then return $\perp$
7.    $(c_1, c_2) = \mathsf{E.Decrypt}(\mathsf{Sk}_2, \mathsf{Ct}_2)$
8.    set $m_1 = \mathsf{E.Decrypt}(\mathsf{Sk}_2, c_1)$ and $m_2 = \mathsf{E.Decrypt}(\mathsf{Sk}_2, c_2)$
9.    return $M(m_1||m_2)$
```

Claim 3.5 Indistinguishability of H_4 from H_3. For simplicity we can assume that $\mathcal{A}$ asks only one token query for the TM M computing the function f. The general case can be handled by a standard hybrid argument. By the statistical binding property of Com and by the correctness of E and by definition of $L^i, i = 1, 2$ and statistical soundness of NIWI, with all except negligible probability, there is exactly *one* pair of ciphertexts $(\mathsf{Ct}_1, \mathsf{Ct}_2)$ that encrypts different inner-messages, the challenge ciphertext, and by definition of the experiments, the message M_0 resulting from decrypting Ct_1 recursively using Sk_1 and the message M_1 resulting from decrypting Ct_2 recursively using Sk_2, are such that $f(M_0) = f(M_1)$ where f is the function associated with the token. Thus, we can invoke the security of $\mathsf{i}\mathcal{O}$ to argue the indistinguishability of the two hybrids.

- H_5. This experiment is identical to H_4 except that Ct_1 is set to an encryption of x_1^1. The commitment com is still generated as $\mathsf{com} = \mathsf{Com}((\mathsf{Ct}_1||\mathsf{Ct}_2); r)$ and the randomness r is still used as witness to compute the proof π in the challenge ciphertext $\mathsf{Ct} = (\mathsf{Ct}_1, \mathsf{Ct}_2, \pi)$.

Claim 3.6 Indistinguishability of H_5 from H_4. The indistinguishability of H_5 from H_4 is symmetrical to that of H_3 from H_2.

- H_6. This experiment is identical to H_5 except that any token is changed to be the obfuscation of the machine $T[M, \mathsf{Sk}_1, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{crs}^1, \mathsf{crs}^2, \mathsf{com}]$.

Claim 3.7 Indistinguishability of H_6 from H_5. The indistinguishability of H_6 from H_5 is symmetrical to that of H_4 from H_3.

- H_7. This experiment is identical to H_6 except that the NIWI proof π in the challenge ciphertext $\mathsf{Ct} = (\mathsf{Ct}_1, \mathsf{Ct}_2)$ is computed with respect to the randomness r used to generate com.

Claim 3.8 Indistinguishability of H_7 from H_6. The indistinguishability of H_7 from H_6 is symmetrical to that of H_2 from H_1.

- H_8. This experiment is identical to H_7 except that the commitment com in the public-key is a commitment to $0^{2m(\lambda)}$.

Claim 3.9 Indistinguishability of H_8 from H_7. The indistinguishability of H_8 from H_7 is symmetrical to that of H_8 from H_7.

The indistinguishability of the above hybrid experiments implies the following theorem.

Theorem 3.10 If for $i = 1, 2$ $\mathsf{NIWI}^i = (\mathsf{CRSGen}^i, \mathsf{Prove}^i, \mathsf{Verify}^i)$ is a NIWI proof system for the $\mathcal{NP}$-language L^i, Com is a (perfectly binding) commitment scheme, $\mathsf{E} = (\mathsf{E.Setup}, \mathsf{E.Encrypt}, \mathsf{E.Decrypt})$ is a IND-CPA secure PKE scheme, and $\mathsf{i}\mathcal{O}$ is an iO for TMs with bounded inputs, then the proposed scheme mFE is a selective IND-Secure mFE scheme for bounded messages supporting one merging operation. If in addition $\mathsf{i}\mathcal{O}$ satisfies succinctness and input-specific running time, so mFE does.

mFE for bounded number of merging operations. It is easy to observe that it is trivial to extend the above construction to support a bounded number q of merging operations with parameters (public- and master secret- key, tokens and ciphertexts) of size proportional to q.

3.2 Extension to support messages of unbounded length

To extend the above scheme to support messages of unbounded length we make the following changes, but first we would like to stress that, whereas in the case of bounded messages it is possible to concatenate strings of fixed length without a separator and indeed we often did this making use of the symbol '||', instead in the case of unbounded messages we would need to separate strings of variable length with a separator but with a slight abuse of notation we will continue to use the previous notation, i.e., we will write $z = x||y$ for strings $x, y \in \{0,1\}^\star$ assuming that it is possible to parse z in x and y.

1. We assume a public-coin differing-inputs obfuscation (diO, in short. See Section D) for TMs with unbounded inputs. This is necessary since the TM to be obfuscated have to read inputs of variable length.

2. We assume collision-resistant hash functions (CRHF, in short. See Section A.1) $\mathsf{CRHF} = (\mathsf{Gen}, \mathsf{Hash})$ mapping strings from $\{0,1\}^\star$ to $\{0,1\}^\lambda$.

3. The Setup procedure of the mFE scheme is identical except that an hashing key hk is generated by $\mathsf{Gen}(1^\lambda)$, and com is a commitment to 0^λ.

4. The languages L^i are changed to the following languages L'^i.

   $L'^1 = \{(\mathsf{Ct}_1, \mathsf{Ct}_2) \in \{0,1\}^{2m(\lambda)} : \exists m, r, r_1, r_2 : (\mathsf{Ct}_1 = \mathsf{E.Encrypt}(\mathsf{Pk}_1, m; r_1) \ \wedge \ \mathsf{Ct}_2 = \mathsf{E.Encrypt}(\mathsf{Pk}_2, m; r_2)) \ \vee \ \mathsf{com} = \mathsf{Com}(\mathsf{Hash}(\mathsf{hk}, \mathsf{Ct}_1||\mathsf{Ct}_2); r)\}$.

   $L'^2 = \{(\mathsf{Ct}_1, \mathsf{Ct}_2) \in \{0,1\}^{2m(\lambda)} : \exists c_1, c_2, c_3, c_4, \pi_1, \pi_2, r, r_1, r_2 : (\mathsf{Ct}_1' = \mathsf{E.Encrypt}(\mathsf{Pk}_1, c_1||c_3; r_1) \wedge \mathsf{Ct}_2 = \mathsf{E.Encrypt}(\mathsf{Pk}_2, c_2||c_4; r_2) \ \wedge \ \mathsf{Verify}^1(\mathsf{crs}^1, \mathsf{c}_1||\mathsf{c}_2) = 1 \ \wedge \ \mathsf{Verify}^1(\mathsf{crs}^1, \mathsf{c}_3||\mathsf{c}_4) = 1) \ \vee \ \mathsf{com} = \mathsf{Com}(\mathsf{Hash}(\mathsf{hk}, \mathsf{Ct}_1||\mathsf{Ct}_2); \mathsf{r})\}$.

   Note that L'^1 is relative to $\mathsf{com}, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{hk}$ and L'^2 is relative to $\mathsf{com}, \mathsf{crs}^1, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{hk}$ and the languages are still in $\mathcal{NP}$. As consequence, the NIWI proofs used in the modified scheme will be relative to the latter languages.

5. The machine $T[M, \mathsf{Sk}_1, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{crs}^1, \mathsf{crs}^2, \mathsf{com}]$ is changed to the machine $T'[M, \mathsf{Sk}_1, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{crs}^1, \mathsf{crs}^2, \mathsf{hk}, \mathsf{com}]$ with the obvious modification that the new machine verifies the proofs with respect to the new languages L'^i (and thus, implicitly using the hashing key hk).

Security reduction. Let us now analyze the changes to the security reduction of Section 3.1. Consider the following series of hybrid experiments against a PPT adversary $\mathcal{A}$ attacking the selective IND-Security of the modifed scheme.

- H_0'. Identical to H_0 except that com is a commitment to $\mathsf{Hash}(\mathsf{hk}, (\mathsf{Ct}_1, \mathsf{Ct}_2))$.
- H_1. This experiment is identical to H_0' except that the commitment com in the public-key is a commitment to $\mathsf{Hash}(\mathsf{hk}, (\mathsf{Ct}_1, \mathsf{Ct}_2))$.

  Claim 3.11 Indistinguishability of H_1' from H_0'. Identical to the indistinguishability of H_1 from H_0.

- H_2'. This experiment is identical to H_1 except that the NIWI proof π in the challenge ciphertext $\mathsf{Ct} = (\mathsf{Ct}_1, \mathsf{Ct}_2)$ is computed with respect to the randomness r used to generate com.

  Claim 3.12 Indistinguishability of H_2' from H_1'. Identical to the indistinguishability of H_2 from H_1.

- H_3'. This experiment is identical to H_2' except that Ct_2 is set to an encryption of x_1^2. The commitment com is still generated as $\mathsf{com} = \mathsf{Com}(\mathsf{Hash}(\mathsf{hk}, (\mathsf{Ct}_1, \mathsf{Ct}_2)); r)$ and the randomness r is still used as witness to compute the proof π in the challenge ciphertext $\mathsf{Ct} = (\mathsf{Ct}_1, \mathsf{Ct}_2, \pi)$.

  Claim 3.13 Indistinguishability of H_3' from H_2'. Identical to the indistinguishability of H_3 from H_2.

- H_4'. This experiment is identical to H_3' except that any token is changed to be the obfuscation of the following machine:
  $T_2'[M, \mathsf{Sk}_2, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{crs}^1, \mathsf{crs}^2, \mathsf{hk}, \mathsf{com}]$ that is identical to $T_2[M, \mathsf{Sk}_2, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{crs}^1, \mathsf{crs}^2, \mathsf{hk}, \mathsf{com}]$ except that it verifies the proofs for the new languages L'^i.

  Claim 3.14 Indistinguishability of H_4 from H_3. For simplicity we can assume that $\mathcal{A}$ asks only one token query for a TM M computing the function f. The general case can be handled by a standard hybrid argument. By the the correctness of E, by definition of $L^i, i = 1, 2$, and statistical soundness of NIWI, with all except negligible probability, the set of pairs of ciphertexts $S = \{(\mathsf{Ct}_1', \mathsf{Ct}_2')\}$ for which (1) there exists an associated valid (i.e., accepted by the verifier) NIWI proof of the fact that $\mathsf{Ct} \in L^2$ (or in the case of ciphertexts of first level, a proof of the fact that $\mathsf{Ct} \in L^1$) and (2) such that the message M_0 resulting from decrypting Ct_1 recursively using Sk_1 and the message M_1 resulting from decrypting Ct_2 recursively using Sk_2 satisfy $f(M_0) \neq f(M_1)$, have one of the following two forms:

  1. $\forall \mathsf{Ct}' = (\mathsf{Ct}_1', \mathsf{Ct}_2') \in S$, $\mathsf{Hash}(\mathsf{hk}, \mathsf{Ct}') = \mathsf{Hash}(\mathsf{hk}, \mathsf{Ct})$ where $\mathsf{Ct} = (\mathsf{Ct}_1, \mathsf{Ct}_2)$ is the value committed in com.
  2. Let $(\mathsf{Ct}_1', \mathsf{Ct}_2') \in S$ and $x_1 = (x_1[L], x_1[R])$ and $x_2 = (x_2[L], x_2[R])$ be the strings resulting from decrypting respectively Ct_1' with Sk_1 and Ct_2' with Sk_2. Then $\mathsf{Hash}(\mathsf{hk}, (x_1[L], x_2[L])) = \mathsf{Hash}(\mathsf{hk}, \mathsf{Ct})$ and $\mathsf{Hash}(\mathsf{hk}, (x_1[R], x_2[R])) = \mathsf{Hash}(\mathsf{hk}, \mathsf{Ct})$, where $\mathsf{Ct} = (\mathsf{Ct}_1, \mathsf{Ct}_2)$ is the value committed in com.

Furthermore, note that the set S does *not* contain the challenge ciphertext Ct committed in com. Consider now the following sampling algorithm $\mathsf{Sampler}$. It takes as input a random string ρ and parses it as (hk, τ). The sampler runs the adversary $\mathcal{A}$ simulating to it the view in experiment H'_4 until $\mathcal{A}$ asks the token query. Specifically, it uses the randomness hk for Hash and the randomness τ to generate the public-/secret- keys for E, for Com, and for the ciphertexts and the NIWI proofs. Then, it outputs the two (non-obfuscated) machines $T'[M, \mathsf{Sk}_1, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{crs}^1, \mathsf{crs}^2, \mathsf{hk}, \mathsf{com}]$ and $T'_2[M, \mathsf{Sk}_2, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{crs}^1, \mathsf{crs}^2, \mathsf{hk}, \mathsf{com}]$. Consider a distinguisher $\mathcal{D}$ that takes as input the randomness ρ and machine M' that is the obfuscation (with respect to $\mathsf{di}\mathcal{O}$) of one of the two previous machines. $\mathcal{D}$ executes all steps of $\mathsf{Sampler}$ and continues the execution of $\mathcal{A}$ answering the token query sending M'. It is easy to see that if $\mathcal{A}$ has non-negligible advantage in distinguishing the two hybrids, so $\mathcal{D}$ does for the two machines. Thus, to prove the claim we have to show that $\mathsf{Sampler}$ is a public-coin differing-inputs sampler. Suppose towards a contradiction that there exists an adversary $\mathcal{B}$ that finds a differing-input to the pair of TMs sampled by $\mathsf{Sampler}$. Then we build an algorithm $\mathcal{C}_{\mathsf{Hash}}$ that breaks the security of Hash. $\mathcal{C}_{\mathsf{Hash}}$ incorporates $\mathsf{Sampler}$ and $\mathcal{B}$. On input a random hashing key hk, the algorithm samples a uniform string τ and runs $\mathcal{B}$ and $\mathsf{Sampler}$ on $\rho = (\mathsf{hk}, \tau)$. Let the output of $\mathsf{Sampler}$ be T', T'_2 and let $\mathsf{Ct}' = (\mathsf{Ct}'_1, \mathsf{Ct}'_2)$ be the output of $\mathcal{B}$. Furthermore, let $\mathsf{Ct} = (\mathsf{Ct}_1, \mathsf{Ct}_2)$ be the challenge ciphertext committed in com that is computed by $\mathsf{Sampler}$ at the beginning of its execution. By the fact that the only distinguishing inputs for T' and T'_2 are strings in S and by correctness of $\mathsf{di}\mathcal{O}$ and by the above two facts, it holds that $\mathsf{Ct}' \neq \mathsf{Ct}$ and either (1) $\mathsf{Hash}(\mathsf{hk}, \mathsf{Ct}') = \mathsf{Hash}(\mathsf{hk}, \mathsf{Ct})$ or (2) $\mathsf{Hash}(\mathsf{hk}, (x'_1[L], x'_2[L])) = \mathsf{Hash}(\mathsf{hk}, \mathsf{Ct})$ and $\mathsf{Hash}(\mathsf{hk}, (x'_1[R], x'_2[R])) = \mathsf{Hash}(\mathsf{hk}, \mathsf{Ct})$, where $x'_1 = (x'_1[L], x'_1[R])$ and $x'_2 = (x'_2[L], x'_2[R])$ are the strings resulting from decrypting respectively Ct'_1 with Sk_1 and Ct'_2 with Sk_2. Observing that in the case (2) $\mathcal{C}_{\mathsf{Hash}}$ has the secret-keys $\mathsf{Sk}_1, \mathsf{Sk}_2$ to decrypt Ct_1 and Ct_2, we conclude that in both cases $\mathcal{C}_{\mathsf{Hash}}$ can find a collision for $\mathsf{Hash}(\mathsf{hk}, \mathsf{Ct})$ as desired.

- H'_5. This experiment is identical to H'_4 except that Ct_1 is set to an encryption of x_1^1. The commitment com is still generated as $\mathsf{com} = \mathsf{Com}(\mathsf{Hash}(\mathsf{hk}, (\mathsf{Ct}_1, \mathsf{Ct}_2)); r)$ and the randomness r is still used as witness to compute the proof π in the challenge ciphertext $\mathsf{Ct} = (\mathsf{Ct}_1, \mathsf{Ct}_2, \pi)$.

Claim 3.15 Indistinguishability of H'_5 from H'_4. The indistinguishability of H'_5 from H'_4 is symmetrical to that of H'_3 from H'_2.

- H'_6. This experiment is identical to H'_5 except that any token is changed to be the obfuscation of the machine $T'[M, \mathsf{Sk}_1, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{crs}^1, \mathsf{crs}^2, \mathsf{hk}, \mathsf{com}]$.

Claim 3.16 Indistinguishability of H'_6 from H'_5. The indistinguishability of H'_6 from H'_5 is symmetrical to that of H'_4 from H'_3.

- H'_7. This experiment is identical to H'_6 except that the NIWI proof π in the challenge ciphertext $\mathsf{Ct} = (\mathsf{Ct}_1, \mathsf{Ct}_2)$ is computed with respect to the randomness r used to generate com.

Claim 3.17 Indistinguishability of H'_7 from H'_6. The indistinguishability of H'_7 from H'_6 is symmetrical to that of H'_2 from H'_1.

- H'_8. This experiment is identical to H'_7 except that the commitment com in the public-key is a commitment to 0^λ.

Claim 3.18 Indistinguishability of H'_8 from H'_7. The indistinguishability of H'_8 from H'_7 is symmetrical to that of H'_8 from H'_7.

The indistinguishability of the above hybrid experiments implies the following theorem.

Theorem 3.19 If for $i = 1, 2$ $\mathsf{NIWI}^i = (\mathsf{CRSGen}^i, \mathsf{Prove}^i, \mathsf{Verify}^\mathsf{i})$ is a NIWI proof system for the $\mathcal{NP}$-language L^i, Com is a (perfectly binding) commitment scheme, $\mathsf{E} = (\mathsf{E.Setup}, \mathsf{E.Encrypt}, \mathsf{E.Decrypt})$ is a IND-CPA secure PKE scheme, Hash is a CRHF, and $\mathsf{di}\mathcal{O}$ is a public-coin differing-inputs obfuscator for TMs (with unbounded inputs), then the modified scheme is a selective IND-Secure mFE scheme for unbounded messages supporting one merging operation. If in addition $\mathsf{i}\mathcal{O}$ satisfies succinctness and input-specific running time, so the mFE scheme does.

4 Future research directions and open problems

The main open problem that we leave to future research or versions of this paper is to provide a formal security reduction for the construction sketched in Section 1.5 or in general the construction of a mFE scheme for unbounded merging operations reducible to reasonable assumptions. Another interesting line of research that our work opens up is the construction of efficient mFE schemes for more restricted functionalities, e.g., extensions of searchable encryption supporting the update of ciphertexts.

References

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *CRYPTO (2)*, pages 500–518, 2013.

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):1301–1328, 2007.

[BDOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522, Interlaken, Switzerland, May 2–6, 2004. Springer, Berlin, Germany.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM Journal on Computing*, 38(5):1661–1694, 2008.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 533–556, 2014.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In

Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Berlin, Germany.

[BIP10] Carlo Blundo, Vincenzo Iovino, and Giuseppe Persiano. Predicate encryption with partial public keys. In Swee-Huay Heng, Rebecca N. Wright, and Bok-Min Goi, editors, *CANS 10: 9th International Conference on Cryptology and Network Security*, volume 6467 of *Lecture Notes in Computer Science*, pages 298–313, Kuala Lumpur, Malaysia, December 12–14, 2010. Springer, Berlin, Germany.

[BLR$^+$14] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. Cryptology ePrint Archive, Report 2014/834, 2014. `http://eprint.iacr.org/`.

[BO13] Mihir Bellare and Adam O'Neill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In *Cryptology and Network Security - 12th International Conference, CANS 2013, Paraty, Brazil, November 20-22. 2013. Proceedings*, pages 218–234, 2013.

[BRS13a] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 461–478. Springer, 2013.

[BRS13b] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private subspace-membership encryption and its applications. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 255–275. Springer, 2013.

[BS14] Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. Cryptology ePrint Archive, Report 2014/550, 2014. `http://eprint.iacr.org/`.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273, Providence, RI, USA, March 28–30, 2011. Springer, Berlin, Germany.

[BSW12] Dan Boneh, Gil Segev, and Brent Waters. Targeted malleability: homomorphic encryption for restricted computations. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 350–366. ACM, 2012.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Berlin, Germany.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222, Interlaken, Switzerland, May 2–6, 2004. Springer, Berlin, Germany.

[CI13] Angelo De Caro and Vincenzo Iovino. On the power of rewinding simulators in functional encryption. *IACR Cryptology ePrint Archive*, 2013:752, 2013.

[CKLM13] Chase, Kohlweiss, Lysyanskaya, and Meiklejohn. Succinct malleable nizks and an application to compact shuffles. In *Theory of Cryptography*, pages 100–119. Springer, 2013.

[DIJ$^+$13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (2)*, volume 8043 of *Lecture Notes in Computer Science*, pages 519–535. Springer, 2013.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 308–317. IEEE Computer Society, 1990.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[GGG$^+$14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602. Springer, 2014.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49. IEEE Computer Society, 2013.

[GGJS13] Shafi Goldwasser, Vipul Goyal, Abhishek Jain, and Amit Sahai. Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/727, 2013. `http://eprint.iacr.org/`.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 465–482, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Berlin, Germany.

[GJKS13] Vipul Goyal, Abhishek Jain, Venkata Koppula, and Amit Sahai. Functional encryption for randomized functionalities. Cryptology ePrint Archive, Report 2013/729, 2013. `http://eprint.iacr.org/`.

[GKL$^+$13] S. Dov Gordon, Jonathan Katz, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. *IACR Cryptology ePrint Archive*, 2013:774, 2013.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 321–340, Singapore, December 5–9, 2010. Springer, Berlin, Germany.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013*, pages 75–92. Springer, 2013.

[HRSV11] Hohenberger, Rothblum, Shelat, and Vaikuntanathan. Securely obfuscating re-encryption. *Journal of Cryptology*, 24(4):694–719, 2011.

[HW15] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 163–172, 2015.

[IPS14] Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. Cryptology ePrint Archive, Report 2014/942, 2014. `http://eprint.iacr.org/`.

[IZ14] Vincenzo Iovino and Karol Zebrowski. Simulation-based secure functional encryption in the random oracle model. Cryptology ePrint Archive, Report 2014/810, 2014. `http://eprint.iacr.org/2014/810`.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing*, STOC '92, pages 723–732, New York, NY, USA, 1992. ACM.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162, Istanbul, Turkey, April 13–17, 2008. Springer, Berlin, Germany.

[KSY14] Ilan Komargodski, Gil Segev, and Eylon Yogev. Functional encryption for randomized functionalities in the private-key setting from minimal assumptions. Cryptology ePrint Archive, Report 2014/868, 2014. `http://eprint.iacr.org/`.

[LMSV12] Loftus, May, Smart, and Vercauteren. On cca-secure somewhat homomorphic encryption. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 55–72. Springer Berlin Heidelberg, 2012.

[LOS$^+$10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91, French Riviera, May 30 – June 3, 2010. Springer, Berlin, Germany.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

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

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 591–608, Cambridge, UK, April 15–19, 2012. Springer, Berlin, Germany.

[PR08] Manoj Prabhakaran and Mike Rosulek. Homomorphic encryption with CCA security. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 667–678, Reykjavik, Iceland, July 7–11, 2008. Springer, Berlin, Germany.

[SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 457–473. Springer, Berlin, Germany, March 15–17, 2009.

[Wat12] Brent Waters. Functional encryption for regular languages. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 218–235, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Berlin, Germany.

[Wat14] Brent Waters. A punctured programming approach to adaptively secure functional encryption. *IACR Cryptology ePrint Archive*, 2014:588, 2014.

A Standard Notions

A.1 Collision-resistant Hash Functions

Definition A.1 [Collision-resistant Hash Functions] We say that a pair of PPT algorithms $(\mathsf{Gen}, \mathsf{Hash})$ is *collision-resistant hash function* (CRHF in short) if:

- $\mathsf{Gen}(1^\lambda)$ outputs an *hashing key* hk. We require that hk is uniformly distributed in $\{0,1\}^\lambda$, i.e., Gen outputs its own random coins. (Note that due to this requirement we could get rid of Gen completely, but for stylistic reasons we stick with such notation).
- There exists some polynomial $l(\lambda)$ such that Hash on input 1^λ and $x \in \{0,1\}^\star$ outputs a string $y \in \{0,1\}^{l(\lambda)}$. If $H(s, \cdot)$ is only defined for inputs x of length $l'(\lambda)$, where $l'(\lambda) > l(\lambda)$, we say that $(\mathsf{Gen}, \mathsf{Hash})$ is a fixed-length collision-resistant hash function for inputs of length l'.
- It holds that for any (possibly, non-uniform) PPT adversary $\mathcal{A}$ the probability of winning in the following game is negligible in λ:

1. $s \leftarrow \mathsf{Gen}(1^\lambda)$;
2. $(x_0, x_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{hk})$;
3. **Output:** $\mathcal{A}$ wins iff $\mathsf{Hash}(\mathsf{hk}, x_0) = \mathsf{Hash}(\mathsf{hk}, x_1)$ and $x_0 \neq x_1$.

A.2 IND-CPA secure PKE

An IND-CPA (or semantically) secure Public Key Encryption (PKE) scheme consists of three PPT algorithms $(\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ described as follows.

- $\mathsf{Setup}(1^\lambda)$: On input 1^λ, it outputs public key Pk and decryption key Sk.
- $\mathsf{Encrypt}(m, \mathsf{Pk})$: On input message m and the public key, it outputs a ciphertext Ct.
- $\mathsf{Decrypt}(\mathsf{Ct}, \mathsf{Sk})$: On input a ciphertext Ct and the decryption key, it outputs m.

The PKE scheme is said to be IND-CPA (or semantically) secure if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that the following is satisfied for any two messages m_0, m_1 and for $b \in \{0,1\}$: $|\Pr\left[\mathcal{A}(1^\lambda, \mathsf{Encrypt}(m_0, \mathsf{Pk})) = b\right] - \Pr\left[\mathcal{A}(1^\lambda, \mathsf{Encrypt}(m_1, \mathsf{Pk})) = b\right]| \leq \mathsf{negl}(\lambda)$.

A.3 Commitment schemes

Definition A.2 [Commitment Schemes] A commitment scheme Com is a PPT algorithm that takes as input a string x and randomness r and outputs $\mathsf{com} \leftarrow \mathsf{Com}(x; r)$. A perfectly binding commitment scheme must satisfy the following properties:

- Perfectly Binding: This property states that two different strings cannot have the same commitment. More formally, $\forall x_1 \neq x_2$ and $r_1, r_2, \mathsf{Com}(x_1; r_1) \neq \mathsf{Com}(x_2; r_2)$.
- Computational Hiding: For all strings x_0 and x_1 (of the same length), for all PPT adversaries $\mathcal{A}$, we have that: $|\Pr[\mathcal{A}(\mathsf{Com}(x_0)) = 1] - \Pr[\mathcal{A}(\mathsf{Com}(x_1)) = 1)]| \leq \mathsf{negl}(k)$.

B NIWI proof systems

We employ non-interactive witness indistinguishability (NIWI) proof systems [FLS90].

Definition B.1 [Non-interactive Proof System]. A non-interactive proof system for a language L with a PPT relation R is a tuple of algorithms $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ such that the following properties hold:

- Perfect Completeness: For every $(x, w) \in R$, it holds that $\Pr[\mathsf{Verify}(\mathsf{crs}, \mathsf{x}, \mathsf{Prove}(\mathsf{crs}, \mathsf{x}, \mathsf{w})) = 1] = 1$, where $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k)$, and the probability is taken over the coins of $\mathsf{CRSGen}, \mathsf{Prove}$ and Verify.
- Statistical Soundness: For every adversary $\mathcal{A}$, it holds that

  $\Pr\left[\mathsf{Verify}(\mathsf{crs}, \mathsf{x}, \pi) = 1 \wedge \mathsf{x} \notin \mathsf{L} | \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\mathsf{k})); (\mathsf{x}, \pi) \leftarrow \mathcal{A}(\mathsf{crs})\right] = \mathsf{negl}(k)$.

We also assume that CRSGen outputs its own random coins, i.e., the procedure output a common *random* string.

Definition B.2 [NIWI]. We say that a non-interactive proof system $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ for a language L with a PPT relation R is witness-indistinguishable if for any triplet (x, w_0, w_1) such that $(x, w_0) \in R$ and $(x, w_1) \in R$, the distributions $\{\mathsf{crs}, \mathsf{Prove}(\mathsf{crs}, x, w_0)\}$ and $\{\mathsf{crs}, \mathsf{Prove}(\mathsf{crs}, x, w_1)\}$ are computationally indistinguishable, where $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k)$.

We remark that for our construction supporting a single merging operation, NIWI proof systems for theorems of ounded size suffice.

C Indistinguishability obfuscation

Definition C.1 [Indistinguishable Obfuscators (iO) for Turing machines] A uniform PPT machine $\mathsf{i}\mathcal{O}$ is called a Turing machine indistinguishable Obfuscators for the Turing machine family $\mathcal{M} = \{\mathcal{M}_n\}$, if the following conditions are satisfied:

- Correctness: $\forall n, \forall M \in \mathcal{M}_n, \forall x \in \{0,1\}^\star$ we have $\Pr[\, M'(x) = M(x) : M' \leftarrow \mathsf{i}\mathcal{O}(1^n, M)\,] = 1$.
- Security of indistinguishability obfuscation: For any (not necessarily uniform) PPT distinguisher $\mathcal{D}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds: For all security parameters $n \in \mathbb{N}$, for all $M_0, M_1 \in \mathcal{M}_n$, aux such that for all $x, M_0(x) = M_1(x)$ and $\mathsf{steps}(M_0, x) = \mathsf{steps}(M_1, x)$ we have that $|\Pr[\,\mathcal{D}(\mathsf{i}\mathcal{O}(1^n, M_0), \mathsf{aux}) = 1\,] - \Pr[\,\mathcal{D}(\mathsf{i}\mathcal{O}(1^n, M_1), \mathsf{aux}) = 1\,]| \leq \mathsf{negl}(n)$.
- Succinctness and input-specific running time: there exists a (global) polynomial $s'(\cdot)$ such that for all $n \in \mathbb{N}$, for all $M \in \mathcal{M}_n$, for all $M_0 \leftarrow \mathsf{i}\mathcal{O}(1^n, M)$, and for all $x \in \{0,1\}^\star$, $\mathsf{steps}(M', x) \leq s'(n, \mathsf{steps}(M, x))$.

We say that an iO is for TMs with bounded inputs if it is an iO for the class of TMs that read inputs of bounded size.

D Public-coin differing-inputs obfuscation

Definition D.1 [Public-Coin Differing-Inputs Sampler for TMs][IPS14]. An efficient non-uniform sampling algorithm $\mathsf{Sampler} = \{\mathsf{Sampler}_n\}$ is called a public-coin differing-inputs sampler for the parameterized collection of TMs $\mathcal{M} = \{\mathcal{M}_n\}$ if the output of $\mathsf{Sampler}_n$ is always a pair of Turing machines $(M_0, M_1) \in \mathcal{M}_n \times \mathcal{M}_n$ such that $|M_0| = |M_1|$ and for all efficient non-uniform (attacker) algorithms $\mathcal{A} = \{\mathcal{A}_n\}$ there exists a negligible function ϵ such that for all $n \in \mathbb{N}$:

$$\Pr\left[\begin{array}{lcl} M_0(x) \neq M_1(x) \wedge & : & (M_0, M_1) \leftarrow \mathsf{Sampler}_n(r), \\ \mathsf{steps}(M_0, x) = \mathsf{steps}(M_1, x) = t & & (x, 1^t) \leftarrow \mathcal{A}_n(r) \end{array}\right] \leq \epsilon(n).$$

By requiring $\mathcal{A}_n$ to output 1^t, we rule out all inputs x for which M_0, M_1 may take more than polynomial steps.

Definition D.2 [Public-Coin Differing-Inputs Obfuscator for TMs]. A uniform PPT algorithm $\mathsf{di}\mathcal{O}$ is a public-coin differing-inputs obfuscator for the parameterized collection of TMs $\mathcal{M} = \{\mathcal{M}_n\}$ if the following requirements hold:

- Correctness: $\forall n, \forall M \in \mathcal{M}_n, \forall x \in \{0,1\}^\star$ we have $\Pr[\, M'(x) = M(x) : M' \leftarrow \mathsf{di}\mathcal{O}(1^n, M)\,] = 1$.

- Security: for every public-coin differing-inputs samplers $\mathsf{Sampler} = \{\mathsf{Sampler}_n\}$ for the collection $\mathcal{M}$, for every efficient non-uniform (distinguishing) algorithm $\mathcal{D} = \{\mathcal{D}_n\}$, there exists a negligible function negl s.t. for all n:

$$|\Pr[\mathcal{D}_n(r, M') = 1 : (M_0, M_1) \leftarrow \mathsf{Sampler}_n(r), M' \leftarrow \mathsf{di}\mathcal{O}(1^n, M_0)]-$$
$$\Pr[\mathcal{D}_n(r, M') = 1 : (M_0, M_1) \leftarrow \mathsf{Sampler}_n(r), M' \leftarrow \mathsf{di}\mathcal{O}(1^n, M_1)]| \leq \mathsf{negl}(n),$$

  where the probability is taken over r and the coins of $\mathsf{di}\mathcal{O}$.

- Succinctness and input-specific running time: there exists a (global) polynomial $s'(\cdot)$ such that for all $n \in \mathbb{N}$, for all $M \in \mathcal{M}_n$, for all $M_0 \leftarrow \mathsf{di}\mathcal{O}(1^n, M)$, and for all $x \in \{0,1\}^\star$, $\mathsf{steps}(M', x) \leq s'(n, \mathsf{steps}(M, x))$.

E FE and its security

Let us now define the notion of a functional encryption scheme FE for a functionality F.

Definition E.1 [Functional Encryption Scheme] A *functional encryption* scheme FE for functionality F is a tuple $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ of 4 algorithms with the following syntax:

- $\mathsf{Setup}(1^\lambda)$: the procedure outputs *public* and *master secret* keys $(\mathsf{Pk}, \mathsf{Msk})$ for *security parameter* λ.

- $\mathsf{KeyGen}(\mathsf{Msk}, k)$: on input a master secret key Msk and *key* $k \in K$, the procedure outputs *token* $\mathtt{Tok}$.

- $\mathsf{Enc}(\mathsf{Pk}, m)$: on input public key Pk and *plaintext* $m \in M$ the procedure outputs *ciphertext* Ct.

- $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok})$: the procedure outputs $y \in \Sigma \cup \{\perp\}$.

In addition we make the following *correctness* requirement: for all $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, all $k \in K$ and $m \in M$, for $\mathtt{Tok} \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k)$ and $\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Pk}, m)$, we have that $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok}) = F(k, m)$ whenever $F(k, m) \neq \perp$, except with negligible probability. (See [BO13] for a discussion about this condition.)

We consider also the following two properties of FE schemes for Turing machines.

- *Succinctness:* A FE scheme for TM is said to be succinct if the ciphertexts have size polynomial in the security parameter and in the message size, and the tokens generated using KeyGen for machine M have size $q(\lambda, |M|)$, where q is a polynomial and $|M|$ is the size of the Turing machine.

- *Input-specific running-time:* A FE scheme for TM is said to have input-specific run time if the decryption algorithm on input token $\mathtt{Tok}$ for machine M and encryption of message m, takes time $p(\lambda, \mathsf{steps}(M, m))$, where p is a polynomial..

Indistinguishability-based security. The indistinguishability-based notion of security for functional encryption scheme
$\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for functionality F defined over (K, M) is formalized by means of the following game $\mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}$ between an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and a *challenger* $\mathcal{C}$. Below, we present the definition for only one message; it is easy to see the definition extends naturally for multiple messages.

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

1. $\mathcal{C}$ generates $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and runs $\mathcal{A}_0$ on input Pk;
2. $\mathcal{A}_0$ submits queries for keys $k_i \in K$ for $i = 1, \ldots, q_1$ and, for each such query, $\mathcal{C}$ computes $\mathtt{Tok}_i = \mathsf{FE.KeyGen}(\mathsf{Msk}, k_i)$ and sends it to $\mathcal{A}_0$.

   When $\mathcal{A}_0$ stops, it outputs two *challenge plaintexts* $m_0, m_1 \in M$ satisfying $|m_0| = |m_1|$ and its internal state $\mathtt{st}$.
3. $\mathcal{C}$ picks $b \in \{0, 1\}$ at random, computes the *challenge ciphertext* $\mathsf{Ct} = \mathsf{Enc}(\mathsf{Pk}, m_b)$ and sends Ct to $\mathcal{A}_1$ that resumes its computation from state $\mathtt{st}$.
4. $\mathcal{A}_1$ submits queries for keys $k_i \in K$ for $i = q_1 + 1, \ldots, q$ and, for each such query, $\mathcal{C}$ computes $\mathtt{Tok}_i = \mathsf{KeyGen}(\mathsf{Msk}, k_i)$ and sends it to $\mathcal{A}_1$.
5. When $\mathcal{A}_1$ stops, it outputs b'.
6. **Output:** if $b = b'$, m_0 and m_1 are of the same length, and $F(k_i, m_0) = F(k_i, m_1)$ for $i = 1 \ldots, q$, then output 1 else output 0.

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

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

Definition E.2 We say that FE is *indistinguishably secure* (IND security, for short) if all probabilistic polynomial-time adversaries $\mathcal{A}$ have at most negligible advantage in the above game.

Simulation-based security In this section, we give a *simulation-based* security definition for FE similar to the one given by Boneh, Sahai and Waters [BSW11]. For simplicity of exposition, below, we present the definition for only one message; it is easy to see the definition extends naturally for multiple messages.

Definition E.3 [Simulation-Based security] A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for functionality F defined over (K, M) is *simulation-secure* (SIM security, for short) if there exists a *simulator* algorithm $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that for all *adversary* algorithms $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ the outputs of the following two experiments are computationally indistinguishable.

```
$\mathsf{RealExp}^{\mathsf{FE},\mathcal{A}}(1^\lambda)$

$(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$;
$(m, \mathsf{aux}) \leftarrow \mathcal{A}_0^{\mathsf{FE.KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk})$;
$\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Pk}, m)$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{FE.KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk}, \mathsf{Ct}, \mathsf{aux})$;
**Output:** $(\mathsf{Pk}, m, \alpha)$
```

```
$\mathsf{IdealExp}_{\mathsf{Sim}}^{\mathsf{FE},\mathcal{A}}(1^\lambda)$

$(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$;
$(m, \mathsf{aux}) \leftarrow \mathcal{A}_0^{\mathsf{FE.KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk})$;
$(\mathsf{Ct}, \mathsf{aux}') \leftarrow \mathsf{Sim}_0(\mathsf{Pk}, |m|, \{k_i, \mathtt{Tok}_{k_i}, F(k_i, m)\}, \tau)$;
$\alpha \leftarrow \mathcal{A}_1^{\mathcal{O}(\cdot)}(\mathsf{Pk}, \mathsf{Ct}, \mathsf{aux})$;
**Output:** $(\mathsf{Pk}, m, \alpha)$
```

Here, $\{k_i\}$ correspond to the token queries of the adversary. Further, oracle $\mathcal{O}(\cdot)$ is the second stage of the simulator, namely algorithm $\mathsf{Sim}_1(\mathsf{Msk}, \mathsf{aux}', \cdot, \cdot)$. Algorithm Sim_1 receives as third argument a key k_j for which the adversary queries a token, and as fourth argument the output value $F(k_j, m)$. Further, note that the simulator algorithm Sim_1 is stateful in that after each invocation, it updates the state aux' which is carried over to its next invocation.