Delegatable Homomorphic Encryption with Applications to Secure Outsourcing of Computation

M. Barbosa[1] and P. Farshim[2]

[1] CCTC/Departamento de Informática, Universidade do Minho,
Campus de Gualtar, 4710-057 Braga, Portugal.
mbb@di.uminho.pt
[2] Information Security Group, Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, United Kingdom.
Pooya.Farshim@rhul.ac.uk

Abstract. In this work we propose a new cryptographic primitive called Delegatable Homomorphic Encryption (DHE). This allows a Trusted Authority to control/delegate the capability to evaluate circuits over encrypted data to untrusted workers/evaluators by issuing tokens. This primitive can be both seen as a public-key counterpart to Verifiable Computation, where input generation and output verification are performed by different entities, or as a generalisation of Fully Homomorphic Encryption enabling control over computations on encrypted data.

Our primitive comes with a series of extra features as follows: 1) there is a one-time setup procedure for all circuits; 2) senders do not need to be aware of the functions which will be evaluated on the encrypted data, nor do they need to register keys; 3) tokens are independent of senders and receiver; and 4) receivers are able to verify the correctness of computation given short auxiliary information on the input data and the function, independently of the complexity of the computed circuit.

We give a modular construction of such a DHE scheme from three components: Fully Homomorphic Encryption (FHE), Functional Encryption (FE), and a (customised) MAC. As a stepping stone, we first define Verifiable Functional Encryption (VFE), and then show how one can build a secure DHE scheme from a VFE and an FHE scheme. We also show how to build the required VFE from a standard FE together with a MAC scheme. All our results hold in the standard model.

Finally, we show how one can build a verifiable computation (VC) scheme generically from a DHE. As a corollary, we get the first VC scheme which remains verifiable even if the attacker can observe verification results.

Keywords. Homomorphism Delegation. Homomorphic Encryption. Functional Encryption. Verifiable Computation. Public-Key Cryptography. Provable Security.

1 Introduction

Consider a public-key encryption scenario where Alice (the sender) and Bob (the receiver) are both resource-constrained. Alice holds message m and Bob is supposed to receive $f(\mathsf{m})$ – for some function f – but neither is able to perform the associated computations. It could be the case that Alice is not even aware of which function is supposed to be applied on m, and Bob might not want to know the details of how that function is implemented. Carol is an untrusted service provider that wishes to sell computing power to Alice and/or Bob. Finally, TA is a trusted authority that is willing to take on the responsibility of approving a concrete circuit for f put forth by Carol. When function f is approved, the TA issues to Carol a token[3] TK_f that enables her to compute f on behalf of Alice and Bob, and publishes a short description h_f of the functionality provided by f to let Alice and Bob know that this is the case.

When Alice encrypts a message m to Bob, the following guarantees are expected:

- If Alice and Bob are honest, then no one else will learn anything about m, as expected from a normal public-key encryption scheme (we call this property *input/output privacy*). Note that this should be the case even if the TA is honest-but-curious.
- If the TA is honest, then no one except Alice will learn anything about m, except what is leaked by f (we call this property *evaluation security*). Note that this should be the case even if Bob and Carol collude.

[3] Our use of the term *token* in this paper aims to distinguish secret keys associated with functions, from those associated with users.

– Furthermore, if the TA is honest, then a successful decryption by Bob assures him that the recovered value was indeed correctly computed as $f(\mathsf{m})$ (we call this property *verifiability*).

In all cases Carol is considered to be potentially dishonest and colluding with other dishonest service providers.

For concreteness, we can picture Bob as managing the IT infrastructure in a company, and wishing to outsource e-mail filtering services to Carol. Ideally, Alice should be able to encrypt her messages to Bob, regardless of the e-mail filtering operations that will take place. Furthermore, Carol should be able to perform e-mail filtering without requiring any pre-processing assistance from Bob: Alice could even pass encrypted messages directly to Carol. Here Bob is not interested in the details of the e-mail filtering algorithm (which might not be public) and need only specify it using a short and high-level description. Furthermore, Bob will be satisfied with assurance from the TA that function f computed by Carol indeed satisfies this specification. In a natural generalisation of this scenario, multiple service providers offer different e-mail filtering algorithms (or other data processing functionalities) and are all competing for Bob's preference.

In the example above, Alice could be completely oblivious of the functionality computed over her encrypted data, but this may not always be the case. In a different scenario, Bob could work for a research institute collecting health-related statistical information. Alice, working in an hospital, would be willing to collaborate with Bob and send him an encrypted database with patients' health records. However, she will only do so if she is assured that only the outputs of certain statistical measures are provided to Bob. Since neither Alice nor Bob possess the computational resources to carry out the necessary computations, they need to resort to Carol. In this case, Alice will need to rely on the TA to guarantee that tokens are only issued for functions that implement statistical measures that do not leak sensitive information about patients. Obviously, input/output confidentiality is an additional requirement of central importance in this setting. Furthermore, Bob's results will only be trustworthy if he can verify that Carol extracted relevant information correctly. This is also a necessary condition to assure Bob that Carol is not cheating and that she is spending the charged resources on evaluating real statistical data.

In this paper we propose and realise a new cryptographic primitive that we call Delegatable Homomorphic Encryption (DHE). The DHE primitive targets the scenarios described above, providing the required functionality and security features that permit satisfying the trust relations between Alice, Bob and Carol. This new primitive can be seen as extensions of a number of previously known cryptographic primitives: a) as a public-key counterpart to Verifiable Computation [7]; b) as a generalisation of Fully Homomorphic Encryption [8] (FHE) that enables control over the Evaluation operation; and c) as a delegatable (and verifiable) variant of Functional Encryption [13] (FE) which achieves both input and output privacy. Before presenting the DHE concept in more detail, we will explain why none of these primitives (nor trivial combinations thereof) completely solves the problems raised by the scenarios described above.

FULLY HOMOMORPHIC ENCRYPTION. Recent breakthroughs [8] in public-key cryptography have made computing over encrypted data using homomorphic encryption a real technologic possibility. In simplistic terms, FHE allows Alice to encrypt data to Bob in such a way that Carol, knowing only the ciphertext and Bob's public key, is able to compute an arbitrary function over the encrypted plaintext without learning anything about its actual value. Confidentiality is guaranteed, and Alice does not need to be aware of the concrete functions that will be computed. Unfortunately, FHE is not intended to guarantee that Carol computes $f(\mathsf{m})$ correctly, and also does not aim to protect Alice from a collusion between Carol and Bob revealing more than $f(\mathsf{m})$.

VERIFIABLE COMPUTATION. Verifiable Computation [7] is a cryptographic primitive that takes advantage of FHE to address the important problem of computation delegation to an untrusted worker. The goal is for the Client (Bob) to delegate to the Worker (Carol) the non-interactive computation of a pre-specified function f with both confidentiality and verifiability guarantees. Bob is resource-constrained and yet should be able to: 1) prepare new inputs to the system; and 2) verify that f was correctly computed over them. This means that input preparation and output verification should involve a much lower computational cost than that required required to compute f unilaterally. Also, the computational load imposed on Carol should not be prohibitive when compared to that required to com-

pute $f(\mathsf{m})$ in the clear. Finally, Bob should have guarantees that the data over which the computation is performed remains secret. The major difference to the DHE scenario above is that VC was designed as a symmetric primitive where Bob acts as both sender and receiver of the encrypted data. Bob can rely on a Trusted Authority to set-up the system on his behalf, but it is still assumed that Bob must be aware of the function that is being computed by Carol when he prepares new inputs to the system.

FUNCTIONAL ENCRYPTION. The DHE scenario also shares similarities with functional encryption [13]. This primitive is a generalisation of other cryptographic notions, including Predicate Encryption (PE) and Attribute-Based Encryption, where ciphertexts and secret keys are associated with extra information in order to exert fine-grained control over who has access to encrypted data. In the view of FE closest to the DHE scenario [4], Alice hides a message m inside the ciphertext. Decryption is then accessible only to Bob if he holds a token TK_f associated with an arbitrary function f of his choice (mapping bit-strings to bit-strings) issued by a Trusted Authority. When invoked on a valid token TK_f, decryption returns $f(\mathsf{m})$. Bob should learn nothing more about the hidden message than that which is leaked by $f(\mathsf{m})$. Unfortunately, FE also comes short of solving the problems raised by the DHE scenario. The problem is that the roles of Bob and Carol are combined into a single entity, which means that Carol is essentially a trusted party if Bob is honest. Put differently, Bob is forced to either evaluate $f(m)$ through decryption, or he must delegate the decryption operation to Carol himself.

THE DHE ARCHITECTURE. The DHE architecture is similar to the (1-hop) FHE scenario[4], extending it with the ability to control which computations can be carried out over encrypted data, and verifying their results. The DHE architecture is as follows.

- Receivers (Bob) generate and publish public keys that can be used by senders (Alice) to prepare encrypted inputs to the system.
- Many senders can independently encrypt inputs to the same receiver. Encryption is independent of the concrete functions that may be computed over encrypted data, and also of the evaluators (Carol) that may compute it.
- The TA is responsible for assigning computation abilities to evaluators. To enable some evaluator (Carol) to carry out computation of a specific function over encrypted data, the TA provides her with a token. This token depends only on the function that originated it, and is independent of senders and receivers.
- When running encryption, Alice computes some auxiliary information about the input. If securely shared with Bob, this permits achieving verifiability of the computed results (without leaking additional information about the original input). We discuss verifiability in more detail below.
- Finally, it is assumed that Alice and Bob are constrained in terms of computation resources, when compared to the TA and Carol. This means that encryption and decryption should be much cheaper to compute than generating tokens and evaluating functions over encrypted inputs.

Verifiability of a decryption result must be considered in the context of a specific input and a specific function. In the DHE architecture this is captured by providing two additional inputs to decryption that set-up this context:

- The first is a short description h_f of the function f that was supposedly evaluated over the encrypted data. This is published by the Trusted Authority, and essentially binds Carol to evaluating a function f that the TA approved. In practice this short description can be a hash or an identifier of a function that the TA has delegated.
- The second is a short piece of information returned by encryption and transferred securely from Alice to Bob. This auxiliary information should not reveal anything about the input data in addition to that which is revealed by the output of the computation (this is another important distinction to the VC scenario). For verifiability to be meaningful, this information must be transferred authentically. In our work we further assume that the

[4] We consider only the case where functions computed over ciphertexts take a single input. A more general scenario where different ciphertexts (possibly from different senders) can be fed to evaluation could be considered, although it is not clear what verifiability would mean in this case).

auxiliary information is also transferred confidentially (e.g. using a combination of standard signature and encryption schemes), and leave the case where it can be known to evaluators for future work[5].

DHE CONSTRUCTION AND VERIFIABLE FUNCTIONAL ENCRYPTION. FE comes as a natural stepping stone towards realising our notion of a DHE. Indeed, FE already provides a means to control the functions that can be computed over encrypted data, as well as restricting the information that is leaked by decryption. However, as we described above, the features offered by FE fall short of what we require for DHE. To solve this problem, we need to consider the FE setting where decryption is no longer performed by Bob (the receiver), but is carried out on a separate module (Carol). Bob, however, must be able to efficiently check that this decryption was carried out honestly, with some help from Alice. We call this new primitive Verifiable Functional Encryption (VFE).

We construct a VFE from a standard FE scheme. The construction is intuitive – we add redundancy to function inputs and outputs to enable verification – but it is non-trivial in its realisation. Alice encrypts a MAC key k together with the message m. The tokens for a function f are now issued for a new function $f^\star$ which computes $f(\mathsf{m})$ and attaches a MAC of the result under key k. However, the security of the underlying FE scheme together with unforgeability of the MAC does *not* seem to be enough to ensure verifiability. Indeed we need special properties from the MAC scheme in order to prove that the FE scheme is preserving its authentication properties in a meaningful way. Our notion of VFE also allows for a clearer modular presentation of our results and, as we shall see, it is helpful in building VC schemes where input/output privacy is not needed.

Verifiable FE provides only a limited form of input privacy and no output privacy (with respect to Carol and the TA). To obtain a DHE scheme we therefore wrap the VFE scheme inside an FHE layer. This technique is similar to those used in [7,5] where Yao's garbled circuits and cut-and-choose protocols (instead of functional encryption) are used in conjunction with FHE. Note that although in this setting the FHE scheme must support the decryption circuit of the verifiable functional encryption scheme, it is not essential in guaranteeing verifiability. This is where the flexibility of our scheme resides.

SECURE VERIFIABLE COMPUTATION. A DHE scheme can be seen as a generalisation of VC where the Client functionality is now carried out by three different parties: the TA carries out the pre-computation stage, Alice prepares the inputs and Bob recovers and verifies the outputs. Not surprisingly, DHE implies a strong form of VC. In particular, the results in this paper solve two open problems identified in [7]. Firstly, any DHE scheme which is secure in the sense that we propose in this paper can be easily converted into a VC scheme that remains verifiable even if the worker (Carol) can observe the outputs of verification and adaptively choose new inputs. Secondly, if one is willing to waive input/output privacy, our construction of a VFE yields the first VC scheme that does not require FHE to achieve verifiability.

STRUCTURE OF THE PAPER. We start by review the related work in the next section. We fix notation in Section 3 before defining the syntax and three security models for DHEs in Section 4. We introduce verifiable functional encryption in Section 5, and give a generic construction from a standard FE scheme and a (customised) MAC. In Section 6 we build a DHE scheme from a verifiable functional encryption scheme and a fully homomorphic encryption scheme. Relation to verifiable computation is discussed in Section 7. We conclude the paper in Section 8 by discussing the instantiation/efficiency of our construction and providing directions for future research.

2 Related Work

Verifiable Computation was proposed in [7] as a solution to the growing need for cryptographic mechanisms that allow a weak computational device to outsource computing to a more powerful and untrusted computational device[6].

[5] Note that although we do not consider this aspect in this paper, the hint could be anonymous in the sense that it reveals nothing about the sender. Our construction actually achieves this.

[6] We refer the interested reader to [7] for a good description of prior work in cryptography and complexity theory related to VC.

The first proposed VC construction combines FHE with Yao's garbled circuits [14]. Unfortunately, this construction only guarantees confidentiality and verifiability if the system is rebooted after an unsuccessful verification. Chung et al. [5] propose a new VC scheme that improves on the efficiency of the construction in [7]. The central idea in this construction is to use a cut-and-choose mechanism whereby the Client randomly mixes pre-chosen inputs with the fresh input to the computation. Since the function values on the pre-chosen inputs can be pre-computed, the Client can check whether the Worker computed the function correctly on all inputs with high probability. The authors also propose a pipelining solution to minimise the limitation of the original VC scheme, whereby one needs to reboot the system after a cheating worker is first detected. This essentially consists of requiring Client and Worker to maintain a state that they gradually update as several computations are successfully delegated. However, the scheme still falls short of preserving its security properties when the attacker has access to a verification oracle. Our scheme improves on previous VC constructions by preserving verifiability in this scenario. Recently and independently, Benabbas et al. [2] proposed a VC scheme aiming at the same level of security as the one adopted in this paper, but for a restricted class of functions (high degree polynomial functions).

Significant advances have recently taken place in the areas of Functional Encryption and Fully Homomorphic Encryption. In this work we use both of these primitives in a black-box way, and so we only briefly refer the most relevant contributions. The first fully homomorphic encryption scheme, where the evaluation algorithm produces ciphertexts whose size is independent of the evaluated circuit, was presented in [8]. An alternative and conceptually simpler construction was subsequently proposed in [19]. The authors in [18,9] consider efficiency improvements and implementation aspects over the previous FHE constructions.

The view of FE that we adopt in this paper is closely related to attribute-hiding predicate-only Predicate Encryption [11]. Indeed, any PE scheme with these characteristics provides exactly the FE functionality for functions (predicates) outputting a single bit. Not surprisingly, a simple transformation suffices to construct the FE we require from a predicate-only attribute-hiding PE (we return to this issue later in the paper). Fully secure predicate encryption schemes for inner products using the dual system encryption technique over composite-order pairing groups and dual pairing vector spaces have been presented in [13]. Prior solutions, e.g. that in [11], offered only selective security, where the adversary must commit to the target of its attack before seeing the public parameters for the scheme. Recently, a fully secure PE scheme for a larger class of predicates has been proposed in [6]. Under this scheme, any k-CNF or k-DNF predicate, for a fixed k, can be associated to the secret key. We discuss possible instantiations of our construction and their efficiency in our concluding remarks.

RELATION TO YAO'S GARBLED CIRCUITS. One of the questions that may arise when analysing the DHE construction proposed in this paper is whether such a scheme can be constructed by adapting the VC construction from [7]. In a nutshell, that construction works as follows.

- The Client creates a Yao circuit for function f as its public key, keeping the input and output labels as her secret key. This is the pre-processing stage.
- For each new input, a new FHE key pair is created and used to encrypt the subset of input wires that encodes the fresh input x.
- The Worker evaluates the circuit in encrypted form using FHE and returns the encrypted output wires to the Client.
- The Client decrypts the output wires, checks that they are correct, and decodes the output.

Intuitively, one-time verifiability comes from the observation that the output wires come from an exponentially large space, which means that the Worker has a negligible probability of forging a new valid output. FHE enables reusing the Yao circuit multiple times.

An adaptation of this approach to the DHE setting would follow a reasoning along the following lines.

- The Yao circuit is the natural choice for the token to be generated by the TA. It could be provided to the Evaluator either in the clear or in encrypted form. This means that the TA must know the input and output wires for the circuit. Again, these can be provided to the TA in the clear or in encrypted form.

– The sender must know the input wires (in the clear or in encrypted form) in order to encode the input.
– The receiver must know the output wires in the clear in order to decrypt and verify the result. This means that any form of FHE encryption applied in the process must be under a public key for which the receiver knows the corresponding secret key.

One possibility to meet these restrictions would be to share the output wires between TA and receiver and publish the input wires encrypted under the receiver's FHE public key. This would allow the sender to homomorphically encode an input, and also the TA to homomorphically construct the Yao circuit. However, if the Worker knows all of the input wires in encrypted form, this allows him to homomorphically fully open the circuit: it can homomorphically maul the outputs without being detected.

This observation indicates that the VC construction seems to impose a necessary restriction on the use of the Yao circuit in order to obtain verifiability: the Evaluator must never see more than one encoded input for the Yao circuit under the same FHE public key. In conclusion, achieving DHE security using a Yao circuit in this manner seems to imply that input wires are sender-specific and secretly shared with the TA. Similarly, output wires would be receiver-specific and secretly shared with the TA. As a corollary senders should be registered, and tokens will be sender and receiver specific.

SECURITY NOTIONS FOR FUNCTIONAL ENCRYPTION. Recently and independently of this work, Boneh, Sahai and Waters [4] and O'Neill [15] have analysed the subtleties in defining syntax and security of functional encryption schemes. Although we have not carried out an in-depth analysis of these issues, our work can now (a posteriori) be related to the conclusions of the referred works. The syntax we have defined for functional encryption is similar to that adopted in [15]. Furthermore, our definition of security is identical to IND-FULL (or IND-TNA for the non-adaptive case) proposed in the same work, and similar to the indistinguishability notion studied in [4]. A limitation has been identified in this definition, in that the adversary may be too restricted (even totally prevented from making a challenge query) for specific functionalities.

In the above mentioned works [4,15] simulation-based notions of security for functional encryption schemes (which are in some sense more natural) are introduced to overcome the said limitations. It is shown in [15] that for *pre-image samplable* functions the simulation-based and indistinguishability-based notions are equivalent under non-adaptive token extraction attacks. Our use of n-key-chameleon MACs relates to this result in the following ways. First, this is deemed necessary by our need to rely on the FE scheme to preserve security properties of the MAC circuit associated with the secret key, which as discussed in [4] raises subtle problems. Second, it allows us to employ a proof-technique similar to that used in [15] to reduce the security of our VFE construction to that of the underlying FE scheme. Finally, our extended functionalities, which include a MAC tag to ensure verifiability, actually *preserve* the pre-image samplability of the original function. This in particular means that equivalence to simulation-based notions can be considered for the resulting DHE in the same terms as for the underlying FE. We leave extending our results to directly deal with simulation-based notions as a direction for future work.

i-HOP HOMOMORPHIC ENCRYPTION. It is interesting to see how our DHE notion relates to i-hop homomorphic encryption as proposed in [10]. There, the main concern is whether one can keep computing on evaluated ciphertexts. In other words, an i-hop homomorphic encryption allows one to call the evaluation procedure on its own output up to i times, while still being able to decrypt the result. The authors in [10] use bootstrapping techniques to convert any (1-hop) homomorphic encryption scheme into an i-hop scheme preserving the function-privacy and/or compactness of the underlying schemes. In DHE we do not (yet) consider function privacy, and our aim is to restrict computation over encrypted data to specific functions in a verifiable manner. To meet these requirements, a DHE scheme must actually ensure that verifiable computation is restricted to one hop. In other words, any Evaluator that is able to convince the receiver to accept the result of composing two (authorised) computations would contradict verifiability in our model. We leave the study of i-hop DHE schemes for future research.

A PARTIAL SOLUTION FROM HOMOMORPHIC SIGN-THEN-ENCRYPT. Homomorphic signatures allow a signer to produce signatures over messages so that any one is able to compute a function f on these messages and a new signature. The new signature authenticates, not only the result (i.e. that it was computed starting from correctly signed data) but also the *correctness* of the computation (i.e. the signature is also bound to the specific circuit f computed over the signed messages). Recent developments in homomorphic signature schemes [3] have produced solutions that can deal with reasonably complex circuits, which permit considering a partial solution to the DHE problem. Indeed, a Sign-Then-Encrypt construction using FHE and homomorphic signatures would allow satisfying the confidentiality and verifiability requirements in the DHE primitive. The evaluator would homomorphically compute both the result and the signature over the encrypted data and, upon decryption, the receiver could take advantage of the correctness assurance provided by the homomorphic signature scheme to verify the result. Unfortunately, such a construction does not provide any protection to the sender if the receiver and the evaluator collude: both will learn everything about the encrypted data, as the receiver's secret key reveals the originally encrypted plaintext.

3 Notation

We write $\mathsf{x} \leftarrow \mathsf{y}$ for assigning value y to variable x. We write $\mathsf{x} \leftarrow_{\$} \mathsf{X}$ for sampling x from set X uniformly at random. If A is a probabilistic algorithm we write $\mathsf{y} \leftarrow_{\$} \mathsf{A}(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ for the action of running A on inputs $\mathsf{x}_1, \ldots, \mathsf{x}_n$ with random coin chosen uniformly at random, and assigning the result to y. For a random variable X we denote by $[\mathsf{X}]$ the support of X, i.e. the set of all values that X takes with non-zero probability. We also denote the empty string by ϵ and use PPT for probabilistic polynomial-time. We say $\eta(\lambda)$ is negligible if $|\eta(\lambda)| \in \lambda^{-\omega(1)}$. Throughout the paper we view functions f as circuits. We call a function space $\mathsf{FunSp}(\mathsf{P})$ and a message space $\mathsf{MsgSp}(\mathsf{P})$, both parameterised by P, compatible if, for any P and for any $f \in \mathsf{FunSp}(\mathsf{P})$, the domain of f is $\mathsf{MsgSp}(\mathsf{P})$.

In this paper we will be using the code-based game-playing language [1]. Each game has an $\mathbf{Initialize}$ and a $\mathbf{Finalize}$ procedure. It also has specifications of procedures to respond to an adversary's various oracle queries. A game Game is run with an adversary $\mathcal{A}$ as follows. First $\mathbf{Initialize}$ runs and its outputs are passed to $\mathcal{A}$. Then $\mathcal{A}$ runs and its oracle queries are answered by the procedures of Game. When $\mathcal{A}$ terminates, its output is passed to $\mathbf{Finalize}$ which returns the outcome of the game y. This interaction is written as $\mathsf{Game}^{\mathcal{A}} \Rightarrow y$. In each game, we restrict attention to legitimate adversaries, which is defined specifically for each game.

4 Delegatable Homomorphic Encryption

We are now ready to formally present the syntax of a DHE scheme.

SYNTAX. A *delegatable homomorphic encryption (DHE) scheme* is specified by six PPT algorithms as follows.

1. $\mathsf{DHE.Setup}(1^{\lambda})$: This is the setup algorithm and is run by the TA. On input a security parameter, it generates a master secret key Msk and a master public key Mpk. We assume Mpk also contains a pair of compatible message and function spaces $\mathsf{DHE.MsgSp}(\mathsf{Mpk})$ and $\mathsf{DHE.FunSp}(\mathsf{Mpk})$.
2. $\mathsf{DHE.Gen}(\mathsf{Mpk})$: This is the key-generation algorithm and is run by the receiver. On input the master public key, it outputs a receiver key-pair $(\mathsf{sk}, \mathsf{pk})$.
3. $\mathsf{DHE.TKGen}(f, \mathsf{Msk})$: This is the token generation algorithm and is run by the TA. On input a (description of a) circuit f and the master key Msk, it generates a token TK for f and a short description of f denoted h_f. For simplicity, we restrict our attention to schemes for which h_f is deterministically generated from f and unique with overwhelming probability (e.g. using a collision-resistant hash function).
4. $\mathsf{DHE.Enc}(\mathsf{m}, \mathsf{pk})$: This is the encryption algorithm and is run by the sender. On input a message m, and a receiver's public key pk it returns a ciphertext c, and some auxiliary information aux.

5. $\mathsf{DHE.Eval}(\mathsf{c}, \mathsf{TK}, \mathsf{pk})$: This is the evaluation algorithm and is run by the evaluator. On input a ciphertext c, a token TK, and a public key pk, it returns a ciphertext $\mathsf{c_{evl}}$.
6. $\mathsf{DHE.Dec}(\mathsf{c_{evl}}, \mathsf{aux}, \mathsf{h}_f, \mathsf{sk})$: This is the deterministic decryption algorithm and is run by the receiver. On input an evaluated ciphertext $\mathsf{c_{evl}}$, auxiliary information aux, a short description of a function h_f, and a secret key sk, it returns either a message m or a special failure symbol $\perp$.

Recall that we assume that the TA publishes the short descriptions h_f so that they are publicly available, and that aux is supposed to be transferred securely from Alice to Bob in order to ensure verifiability.

CORRECTNESS. A DHE scheme is correct if for any $\lambda \in \mathbb{N}$, any $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{DHE.Setup}(\mathsf{Mpk})]$, any $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{DHE.Gen}(\mathsf{Mpk})]$, any $\mathsf{m} \in \mathsf{DHE.MsgSp}(1^\lambda)$ and any $f \in \mathsf{DHE.FunSp}(\mathsf{Mpk})$ we have that

$$\Pr\left[\begin{array}{rl} (\mathsf{TK}, \mathsf{h}_f) & \leftarrow_\$ \mathsf{DHE.TKGen}(f, \mathsf{Msk}) \\ (\mathsf{c}, \mathsf{aux}) & \leftarrow_\$ \mathsf{DHE.Enc}(\mathsf{m}, \mathsf{pk}) \\ \mathsf{c_{evl}} & \leftarrow_\$ \mathsf{DHE.Eval}(\mathsf{c}, \mathsf{TK}, \mathsf{pk}) \end{array} : f(\mathsf{m}) = \mathsf{DHE.Dec}(\mathsf{c_{evl}}, \mathsf{aux}, \mathsf{h}_f, \mathsf{sk})\right] = 1.$$

COMPACTNESS. Intuitively, in a compact DHE scheme the evaluation algorithm takes on the computational load of evaluating circuits over the data. This requirement, as in fully homomorphic encryption, rules out trivial constructions where evaluation appends the circuit to ciphertexts, and it is the decryption which bears the computational load. More formally, a DHE scheme is compact if there is a fixed polynomial $\mathrm{B}(\cdot)$ such that for any $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{DHE.Setup}(1^\lambda)]$, any $f \in \mathsf{DHE.FunSp}(\mathsf{Mpk})$, any $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{DHE.Gen}(\mathsf{Mpk})]$, any $\mathsf{m} \in \mathsf{DHE.MsgSp}(\mathsf{Mpk})$, any $(\mathsf{TK}, \mathsf{h}_f) \in [\mathsf{DHE.TKGen}(f, \mathsf{Msk})]$, any $(\mathsf{c}, \mathsf{aux}) \in [\mathsf{DHE.Enc}(\mathsf{m}, \mathsf{pk})]$, and any $\mathsf{c_{evl}} \in [\mathsf{DHE.Eval}(\mathsf{c}, \mathsf{TK}, \mathsf{pk})]$, the size of $(\mathsf{c_{evl}}, \mathsf{aux}, \mathsf{h}_f, \mathsf{sk})$ is at most $\mathrm{B}(\lambda + |f(\mathsf{m})|)$ (independently of the size of f).

OUTSOURCEABILITY. Roughly speaking, this requirement rules out schemes which pre-compute functions at the encryption stage. Formally, a DHE scheme is outsourceable if for any $c > 0$ and sufficiently large λ, we have that for any $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{DHE.Setup}(1^\lambda)]$, any $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{DHE.Gen}(\mathsf{Mpk})]$, and any message $\mathsf{m} \in \mathsf{DHE.MsgSp}(\mathsf{Mpk})$, and any $f \in \mathsf{DHE.FunSp}(\mathsf{Mpk})$, the time-complexity of $\mathsf{DHE.Enc}(\mathsf{m}, \mathsf{pk})$ is at most c times the time-complexity of $f(\mathsf{m})$.

INPUT/OUTPUT PRIVACY. We formulate a notion of input/output privacy that requires no party (including an honest-but-curious TA) except possibly the sender or the receiver should learn anything about the data enclosed in DHE ciphertexts. Formally, the TA-IND-CPA security of a DHE scheme requires the advantage of any PPT adversary $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\mathsf{ta\text{-}ind\text{-}cpa}}_{\mathsf{DHE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathsf{TA\text{-}IND\text{-}CPA}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}\right] - 1,$$

to be negligible, where game TA-IND-CPA is shown in Figure 1. Note that the adversary receives Msk, which enables it to extract tokens for *all* functions. Also note that the adversary is not given the auxiliary information aux that is used to ensure verifiability. This is consistent with the assumption in this model that the sender and the receiver are trusted.

proc. Initialize(λ):	**proc. Left-Right**$(\mathsf{m}_0, \mathsf{m}_1)$:	**proc. Finalize**(b'):
$b \leftarrow_\$ \{0,1\}$	$(\mathsf{c}, \mathsf{aux}) \leftarrow_\$ \mathsf{DHE.Enc}(\mathsf{m}_b, \mathsf{pk})$	Return $(b = b')$
$(\mathsf{Msk}, \mathsf{Mpk}) \leftarrow_\$ \mathsf{DHE.Setup}(1^\lambda)$	Return c	
$(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{DHE.Gen}(\mathsf{Mpk})$		
Return $(\mathsf{Msk}, \mathsf{Mpk}, \mathsf{pk})$		

Fig. 1: Game defining the TA-IND-CPA security of a DHE scheme. An adversary is legitimate if it calls **Left-Right** exactly once on two messages of equal length.

VERIFIABILITY. Verifiability for DHEs essentially requires an evaluator to be unable to convince the sender and the receiver (sharing a secret aux that contextualises a concrete input) that it has honestly computed an incorrect value. We capture this by allowing the adversary to obtain a polynomial number of tokens for functions of its choice. We also allow the attacker to obtain a polynomial number of encrypted inputs for messages of her choice. The goal of the adversary is then to produce a ciphertext that is accepted and decrypts to an incorrect value. The model is strengthened by introducing a decryption oracle that captures the possibility that the evaluator observes the results of decryptions (verifications) executed by the receiver. More formally, the VRF-CCAx security of a DHE scheme for $\mathsf{x} \in \{1, 2\}$ requires the advantage of any PPT adversary $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\mathsf{vrf\text{-}cca x}}_{\mathsf{DHE},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{VRF\text{-}CCAx}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}\right]$$

to be negligible, where game VRF-CCAx is shown in Figure 2.

proc. Initialize(λ):	**proc. Decrypt**$(\mathsf{c}, i, \mathsf{h}_f)$:	**proc. Finalize**$(\mathsf{c}^\star, i, \mathsf{h}_f)$:
$\mathsf{List} \leftarrow \{\}$; $\mathsf{TKList} \leftarrow \{\}$; $i \leftarrow 0$	Let $(\mathsf{m}, \mathsf{aux})$ be s.t. $(i, \mathsf{m}, \mathsf{aux}) \in \mathsf{List}$	If $(i, \star, \star) \notin \mathsf{List}$ Return F
$(\mathsf{Msk}, \mathsf{Mpk}) \leftarrow_{\$} \mathsf{DHE.Setup}(1^\lambda)$	$\mathsf{m} \leftarrow \mathsf{DHE.Dec}(\mathsf{c}, \mathsf{aux}, \mathsf{h}_f, \mathsf{sk})$	If $(\star, \mathsf{h}_f) \notin \mathsf{TKList}$ Return F
$(\mathsf{sk}, \mathsf{pk}) \leftarrow_{\$} \mathsf{DHE.Gen}(\mathsf{Mpk})$	Return m	Let $(\mathsf{m}, \mathsf{aux})$ be s.t. $(i, \mathsf{m}, \mathsf{aux}) \in \mathsf{List}$
Return $(\mathsf{Mpk}, \mathsf{pk})$		Let (f, h_f) be s.t. $(f, \mathsf{h}_f) \in \mathsf{TKList}$
	proc. Encrypt(m):	$\mathsf{m}^\star \leftarrow \mathsf{DHE.Dec}(\mathsf{c}^\star, \mathsf{aux}, \mathsf{h}_f, \mathsf{sk})$
proc. Token(f):	$(\mathsf{c}, \mathsf{aux}) \leftarrow_{\$} \mathsf{DHE.Enc}(\mathsf{m}, \mathsf{pk})$	If $\mathsf{m}^\star = \perp$ Return F
$(\mathsf{TK}, \mathsf{h}_f) \leftarrow_{\$} \mathsf{DHE.TKGen}(f, \mathsf{Msk})$	$i \leftarrow i + 1$; $\mathsf{List} \leftarrow \mathsf{List} \cup \{(i, \mathsf{m}, \mathsf{aux})\}$	Return $(\mathsf{m}^\star \neq f(\mathsf{m}))$
$\mathsf{TKList} \leftarrow \mathsf{TKList} \cup \{(f, \mathsf{h}_f)\}$	Return c	
Return $(\mathsf{TK}, \mathsf{h}_f)$		

Fig. 2: Game defining the VRF-CCAx security of a DHE scheme. An adversary is legitimate if: 1) it calls $\mathbf{Decrypt}$ with an i such that $(i, \star, \star) \in \mathsf{List}$; and 2) it does not call $\mathbf{Token}$ after calling $\mathbf{Encrypt}$ when $\mathsf{x} = 1$.

A natural question that arises after introducing the previous models is how does verifiability relate to IND-CPA security. We clarify this issue in Appendix A by introducing an IND-CCA security model for DHEs. In this model, where TA, Alice and Bob are assumed to be honest, the decryption oracle mimics the actions of Bob, knowing the correct secret aux for the challenge ciphertext and the set of valid function descriptions published by the TA. We show that verifiability and I/O privacy jointly imply IND-CCA security. Note that this does not mean that the DHE retains its I/O privacy in the presence of a verification oracle: the decryption oracle in the CCA security model does not allow the adversary to infer the verification result from the oracle output.

Another important feature of DHE schemes is that they provide a reduced level of security to Alice even when Carol and Bob collude. We consider this situation next.

EVALUATION SECURITY. Evaluation security aims to guarantee senders that no information about encrypted messages beyond that which can be obtained from evaluations and decryptions for authorised functions is leaked. This means that, even if an evaluator and a receiver collude (note that the attacker gets to see aux and the receiver's secret key) they cannot obtain more information about the original message than they should. In addition to a $\mathbf{Left\text{-}Right}$ oracle, the adversary gets a token extraction oracle to which it can only query functions that do not allow for trivial distinguishing attacks. Furthermore, the adversary may observe evaluations under other functions for all ciphertexts except the challenge. Under this notion of security, a collusion of evaluators and receivers cannot misuse their resources to compute unauthorised functions. In other words the scheme provides *resource protection*. Formally, the IND-EVALx security of a DHE scheme for $\mathsf{x} \in \{1, 2\}$ requires the advantage of any PPT adversary $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\mathsf{ind\text{-}eval x}}_{\mathsf{DHE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathsf{IND\text{-}EVALx}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}\right] - 1$$

to be negligible, where game IND-EVALx is shown in Figure 3.

proc. Initialize(λ):
$b \leftarrow_{\$} \{0,1\}$; $\mathsf{List} \leftarrow \{\}$; $\mathsf{TKList} \leftarrow \{\}$
$(\mathsf{Msk}, \mathsf{Mpk}) \leftarrow_{\$} \mathsf{DHE.Setup}(1^{\lambda})$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow_{\$} \mathsf{DHE.Gen}(\mathsf{Mpk})$
Return $(\mathsf{Mpk}, \mathsf{sk}, \mathsf{pk})$

proc. Token(f):
$(\mathsf{TK}, \mathsf{h}_f) \leftarrow_{\$} \mathsf{DHE.TKGen}(f, \mathsf{Msk})$
$\mathsf{TKList} \leftarrow \mathsf{TKList} \cup \{f\}$
Return $(\mathsf{TK}, \mathsf{h}_f)$

proc. Evaluate(c, f):
$(\mathsf{TK}, \mathsf{h}_f) \leftarrow_{\$} \mathsf{DHE.TKGen}(f, \mathsf{Msk})$
$\mathsf{c}_{\mathsf{evl}} \leftarrow_{\$} \mathsf{DHE.Eval}(\mathsf{c}, \mathsf{TK}, \mathsf{pk})$
Return $\mathsf{c}_{\mathsf{evl}}$

proc. Left-Right$(\mathsf{m}_0, \mathsf{m}_1)$:
$(\mathsf{c}, \mathsf{aux}) \leftarrow_{\$} \mathsf{DHE.Enc}(\mathsf{m}_b, \mathsf{pk})$
$\mathsf{List} \leftarrow \mathsf{List} \cup \{(\mathsf{c}, \mathsf{aux})\}$
Return $(\mathsf{c}, \mathsf{aux})$

proc. Finalize(b'):
Return $(b = b')$

Fig. 3: Game defining the IND-EVALx security of a DHE scheme. An adversary is legitimate if: 1) it calls **Left-Right** exactly one on two messages of equal length; 2) it never queries **Token** with an f such that $f(\mathsf{m}_0) \neq f(\mathsf{m}_1)$; 3) it never calls **Evaluate** with a c such that $(\mathsf{c}, \star) \in \mathsf{List}$; and 4) it does not call **Evaluate** or **Token** after calling **Left-Right** when $\mathsf{x} = 1$.

5 Verifiable Functional Encryption

As a stepping stone towards realising our DHE notion, we first introduce Verifiable Functional Encryption (VFE). Our discussion is based on the definitions of standard functional encryption presented in Appendix B.3.

SYNTAX. The syntax of a VFE scheme is similar to that of an FE scheme with the caveat that additional parameters are returned by the encryption, decryption and token generation algorithms that can be fed to a new verification algorithm. The verification algorithm allows anyone who is given an output of decryption to verify that it was honestly computed with the additional help of contextual information that binds the check to a specific encryption operation and a specific function that was supposedly computed over the encrypted data. Concretely, a *verifiable functional encryption (VFE) scheme* is defined through the following PPT algorithms.

1. $\mathsf{VFE.Setup}(1^{\lambda})$: This is the setup algorithm. On input a security parameter 1^{λ}, it outputs a master public key Mpk, and a master secret key Msk. We assume Mpk also contains a pair of compatible message and function spaces $\mathsf{VFE.MsgSp}(\mathsf{Mpk})$ and $\mathsf{VFE.FunSp}(\mathsf{Mpk})$.
2. $\mathsf{VFE.TKGen}(f, \mathsf{Msk})$: This is the token generation algorithm. On input the master secret key Msk and a function f, it outputs a token TK and a short identifier h_f for f. For simplicity, we restrict our attention to schemes for which these identifiers are deterministically generated from f and unique with overwhelming probability (e.g. using a collision-resistant hash function).
3. $\mathsf{VFE.Enc}(\mathsf{m}, \mathsf{Mpk})$: This is the verifiable encryption algorithm. On input a message m, it outputs a ciphertext c and a key k. The returned key will allow a sender to verify an output of decryption using $\mathsf{VFE.Ver}$.
4. $\mathsf{VFE.Dec}(\mathsf{c}, \mathsf{TK}, \mathsf{Mpk})$: This is the deterministic decryption algorithm. On input a token TK, a ciphertext c, and a master key Mpk, it outputs a range point/tag pair (y, t), or a special failure symbol $\perp$.
5. $\mathsf{VFE.Ver}((y, \mathsf{t}), \mathsf{h}_f, \mathsf{k}, \mathsf{Mpk})$: This is the deterministic verification algorithm. On input a range point/tag pair (y, t), a fingerprint value h_f and keys k and Mpk, it returns a decision value T or F. We also require the verification algorithm to accept only one tag for each $(y, \mathsf{h}_f, \mathsf{k})$ input.

CORRECTNESS. A VFE scheme is correct if for any $\lambda \in \mathbb{N}$, any $(\mathsf{Mpk}, \mathsf{Msk}) \in [\mathsf{VFE.Setup}(1^{\lambda})]$, any $f \in \mathsf{VFE.FunSp}(\mathsf{Mpk})$, and any $\mathsf{m} \in \mathsf{VFE.MsgSp}(\mathsf{Mpk})$, we have that

$$\Pr\left[\begin{array}{rl} (\mathsf{TK}, \mathsf{h}_f) & \leftarrow_{\$} \mathsf{VFE.TKGen}(f, \mathsf{Msk}) \\ (\mathsf{c}, \mathsf{k}) & \leftarrow_{\$} \mathsf{VFE.Enc}(\mathsf{m}, \mathsf{Mpk}) \\ (y, \mathsf{t}) & \leftarrow_{\$} \mathsf{VFE.Dec}(\mathsf{c}, \mathsf{TK}, \mathsf{Mpk}) \end{array} : f(\mathsf{m}) = y \wedge \mathsf{VFE.Ver}((y, \mathsf{t}), \mathsf{h}_f, \mathsf{k}, \mathsf{Mpk}) = \mathsf{T}\right] = 1.$$

OUTSOURCEABILITY. A VFE scheme is outsourceable if for every $c > 0$ and sufficiently large λ we have that for any $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{VFE.Setup}(1^{\lambda})]$, any message $\mathsf{m} \in \mathsf{VFE.MsgSp}(\mathsf{Mpk})$, and any $f \in \mathsf{VFE.FunSp}(\mathsf{Mpk})$, the

time-complexity of $\mathsf{VFE.Enc}(\mathsf{m}, \mathsf{Mpk})$ is at most c times the time-complexity of $f(\mathsf{m})$. Put differently, for every f, the time-complexity of $\mathsf{VFE.Enc}(\cdot,\cdot)$ is $o(|f|)$, where $|f|$ denotes the number of gates that f contains (circuit size of f). This requirement rules out trivial constructions of functional cryptosystems which pre-compute f during encryption.

PLAINTEXT SECURITY. The $\mathsf{IND\text{-}CCA}x$ security of a VFE scheme for $x \in \{0,1,2\}$ is defined similarly to that of a standard FE scheme (see Figure 14), with the difference that the $\mathbf{Left\text{-}Right}$ procedure now returns (c, k). Note that this definition essentially requires the scheme to leak nothing about the challenge except $f(\mathsf{m}_b)$ for all f for which the receiver obtains a token, even if the attacker is given k as auxiliary information.

VERIFIABILITY. The intuition here is similar to that in the DHE scenario: correctness of outsourced decryption should be checkable. More precisely, the $\mathsf{VRF\text{-}CCA}x$ security of a VFE scheme for $x \in \{1,2\}$ requires the advantage of any PPT adversary $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\mathsf{vrf\text{-}cca}x}_{\mathsf{VFE},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{VRF\text{-}CCA}x^{\mathcal{A}}_{\mathsf{VFE}}(\lambda) \Rightarrow \mathsf{T}\right],$$

to be negligible, where game $\mathsf{VRF\text{-}CCA}x$ is shown in Figure 4.

proc. Initialize(λ):	**proc. Decrypt**(c, f):	**proc. Finalize**$((y,\mathsf{t}), i, f)$:
$\mathsf{List} \leftarrow \{\}$; $\mathsf{TKList} \leftarrow \{\}$; $i \leftarrow 0$	$(\mathsf{TK}, \mathsf{h}_f) \leftarrow_{\$} \mathsf{VFE.TKGen}(f, \mathsf{Msk})$	If $(i, \star, \star) \notin \mathsf{List}$ Return F
$(\mathsf{Msk}, \mathsf{Mpk}) \leftarrow_{\$} \mathsf{VFE.Setup}(1^\lambda)$	$(y, \mathsf{t}) \leftarrow \mathsf{VFE.Dec}(\mathsf{c}, \mathsf{TK}, \mathsf{Mpk})$	If $(f, \mathsf{h}_f) \notin \mathsf{TKList}$ Return F
Return Mpk	Return (y, t)	Let (m, k) be s.t. $(i, \mathsf{m}, \mathsf{k}) \in \mathsf{List}$
		If $\mathsf{VFE.Ver}((y,\mathsf{t}), \mathsf{h}_f, \mathsf{k}, \mathsf{Mpk}) = \mathsf{F}$
proc. Token(f):	**proc. Encrypt**(m):	Return F
$(\mathsf{TK}, \mathsf{h}_f) \leftarrow_{\$} \mathsf{VFE.TKGen}(f, \mathsf{Msk})$	$(\mathsf{c}, \mathsf{k}) \leftarrow_{\$} \mathsf{VFE.Enc}(\mathsf{m}, \mathsf{Mpk})$	Return $(y \neq f(\mathsf{m}))$
$\mathsf{TKList} \leftarrow \mathsf{TKList} \cup \{(f, \mathsf{h}_f)\}$	$i \leftarrow i+1$; $\mathsf{List} \leftarrow \mathsf{List} \cup \{(i, \mathsf{m}, \mathsf{k})\}$	
Return $(\mathsf{TK}, \mathsf{h}_f)$	Return c	

Fig. 4: Game defining the $\mathsf{VRF\text{-}CCA}x$ security of a VFE scheme. An adversary is legitimate it does not call $\mathbf{Token}$ or $\mathbf{Decrypt}$ after calling $\mathbf{Encrypt}$ when $x = 1$.

5.1 Constructing a Verifiable FE Scheme

ADDING VERIFIABILITY TO FUNCTIONS. In our construction of a VFE we use a transformation which given a pair of compatible spaces $\mathsf{FunSp}(1^\lambda)$ and $\mathsf{MsgSp}(1^\lambda)$ constructs a new function family that enables verifiability without loss of functionality – we will call this the verifiable function family. The intuition behind this construction is simple: we extend any function $f(\mathsf{m})$ to a function $f^\star(\mathsf{m}, \mathsf{k})$ that takes also a secret key and computes $(f(\mathsf{m}), \mathsf{t})$ where t authenticates $f(\mathsf{m})$ under k.

Formally, in order to establish a verification context that is a binding both to a concrete function and a concrete input, we take a collision resistant hash function family (CRH) and a MAC scheme (possibly with a global set-up procedure and a trapdoor). We use CRH to derive function fingerprints from function descriptions, and define

$$f^\star(\mathsf{m}, \mathsf{k}) := (f(\mathsf{m}), \mathsf{MAC.Tag}((f(\mathsf{m}), \mathsf{h}_f), \mathsf{k}, \mathsf{mk})), \text{ where } \mathsf{h}_f \leftarrow \mathsf{CRH.H}(\langle f \rangle, \mathsf{hk}).$$

We note that function identifiers (or fingerprints) will be unique with overwhelming probability, and that one can verify that function $f^\star$ was correctly computed simply by checking that $\mathsf{MAC.Tag}((y, \mathsf{h}_f), \mathsf{k}, \mathsf{mk}) = \mathsf{t}$.

Also note that since MAC and CRH are deterministic the above transformation enjoys the property that $f^\star(\mathsf{m}_0, \mathsf{k}) = f^\star(\mathsf{m}_1, \mathsf{k})$ if and only if $f(\mathsf{m}_0) = f(\mathsf{m}_1)$. Furthermore given $f^\star(\mathsf{m}, \mathsf{k})$ one can read off $f(\mathsf{m})$ in time equal to $|f(\mathsf{m})|$.

THE VFE CONSTRUCTION. Our construction is simple but somewhat surprising in its capabilities. We start with any FE scheme accepting circuits for the verifiable function family that we defined above. We then show that

encrypting a fresh secret key for the MAC along with the input to the function we want to compute is enough to achieve verifiability. Although this technique is intuitive, it has proven to be elusive to formalise and prove secure. It is clear that the MAC scheme must be unforgeable for one to have any hope of proving this result. However, it is not trivial to prove that including a MAC key inside a ciphertext, and issuing tokens for arbitrary functions in the above verifiable function family, does not enable the adversary to forge a new MAC on the same key.

To better see where the problem arises, note that IND-CCA security of the FE scheme guarantees that no information about the input (the message and the MAC key) beyond that leaked through the extracted function(s) is leaked to an adversary. Hence, before one can reduce the verifiability of the FE to the unforgeability of the MAC one must first have a meaningful reduction to the IND-CCA security of the FE, replacing the MAC key with a different one that can be embedded in the challenge ciphertext provided to the verifiability adversary (this is because the real MAC key will not be available in the reduction to unforgeability). The trick is then to find a new key that gives rise to the same MAC, which can be substituted inside the challenge ciphertext without this being perceptible to the adversary. To achieve this result we need a MAC with the following property: it is possible to find a key which leads to identical tags on any n messages, and this key can be given to the adversary without hindering unforgeability. We call a MAC with this property an n-key-chameleon MAC. In Appendix C we provide the precise definitions and give an implementation based on an information-theoretically secure MAC.

Note that this proof-technique requires us to restrict ourselves to adversaries that perform at most n token extraction queries, all before calling the challenge. Fortunately, this is enough to ensure that our results imply a strongly secure VC scheme, as we will see later in the paper. We now formalise this intuition.

Let FE be a (standard) functional encryption scheme such that for any $\lambda \in \mathbb{N}$ and any given compatible spaces $\mathsf{FunSp}(1^\lambda)$ and $\mathsf{MsgSp}(1^\lambda)$, we have that $\mathsf{FE.FunSp}(\mathsf{Mpk})$ includes $f^\star$ as defined in the displayed equation above, for any $f \in \mathsf{FunSp}(1^\lambda)$, any $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{FE.Setup}(1^\lambda)]$, any $\mathsf{hk} \in [\mathsf{CRH.KeySp}(1^\lambda)]$, and any $(\mathsf{td}, \mathsf{mk}) \in [\mathsf{MAC.Setup}(1^\lambda)]$. We define our verifiable functional encryption scheme VFE as shown in Figure 5. Note that our construction satisfies the uniqueness requirement on tags accepted by the verification algorithm by recomputing a deterministic MAC. Security properties of the scheme are stated next, and we refer to Appendices D and E for precise theorem statements and proofs.

Theorem 1 (Informal). *The construction in Figure 5 is IND-CCA if the underlying FE scheme is IND-CCA. It is also verifiable under non-adaptive bounded attacks if the FE scheme is IND-CCA1 and the MAC is unforgeable.*

- $\mathsf{VFE.Setup}(1^\lambda)$
  1. $(\mathsf{Msk}, \mathsf{Mpk}) \leftarrow_\$ \mathsf{FE.Setup}(1^\lambda)$
  2. $(\mathsf{td}, \mathsf{mk}) \leftarrow_\$ \mathsf{MAC.Setup}(1^\lambda)$
  3. $\mathsf{hk} \leftarrow_\$ \mathsf{CRH.Gen}(1^\lambda)$
  4. Return $((\mathsf{Msk}, \mathsf{mk}, \mathsf{hk}), (\mathsf{Mpk}, \mathsf{mk}, \mathsf{hk}))$
- $\mathsf{VFE.TKGen}(f, (\mathsf{Msk}, \mathsf{mk}, \mathsf{hk}))$
  1. $\mathsf{h}_f \leftarrow \mathsf{CRH.H}(\langle f \rangle, \mathsf{hk})$
  2. $f^\star(x_1, x_2) := (f(x_1), \mathsf{MAC.Tag}((f(x_1), \mathsf{h}_f), x_2, \mathsf{mk}))$
  3. $\mathsf{TK} \leftarrow_\$ \mathsf{FE.TKGen}(f^\star, \mathsf{Msk})$
  4. Return $(\mathsf{TK}, \mathsf{h}_f)$
- $\mathsf{VFE.Enc}(\mathsf{m}, (\mathsf{Mpk}, \mathsf{mk}, \mathsf{hk}))$
  1. $\mathsf{k} \leftarrow_\$ \mathsf{MAC.KeySp}(\mathsf{mk})$
  2. $\mathsf{c} \leftarrow_\$ \mathsf{FE.Enc}((\mathsf{m}, \mathsf{k}), \mathsf{Mpk})$
  3. Return (c, k)
- $\mathsf{VFE.Dec}(\mathsf{c}, \mathsf{TK})$
  1. $\mathsf{m} \leftarrow \mathsf{FE.Dec}(\mathsf{c}, \mathsf{TK})$
  2. Parse $(y, \mathsf{t}) \leftarrow \mathsf{m}$
  3. Return (y, t)
- $\mathsf{VFE.Ver}((y, \mathsf{t}), \langle f \rangle, \mathsf{k}, (\mathsf{Mpk}, \mathsf{mk}, \mathsf{hk}))$
  1. Return $\mathsf{MAC.Tag}((y, \mathsf{h}_f), \mathsf{k}, \mathsf{mk}) = \mathsf{t}$

Fig. 5: A VFE scheme based on a standard FE scheme and an n-key-chameleon MAC.

6 A Strongly Secure DHE Scheme

We now present a generic construction of a DHE for a given function space. The construction uses the VFE scheme introduced in the previous section as a building block. The other component in our construction is an FHE scheme. For the definitions and security models of FHE we refer the reader to Appendix B.1.

Let FHE be a compact homomorphic public-key encryption scheme (supporting arity-1 functions), and let VFE be an outsourceable verifiable functional encryption scheme for the function family constructed in the previous section. Suppose also that for any $\lambda \in \mathbb{N}$, any $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{FHE.Gen}(1^\lambda)]$ and any $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{VFE.Gen}(1^\lambda)]$ we have that: $\mathsf{FHE.MsgSp}(\mathsf{pk}) = \mathsf{VFE.CphSp}(\mathsf{Mpk})$ and $\mathsf{VFE.Dec}(\cdot,\cdot) \in \mathsf{FHE.FunSp}(\mathsf{pk})$. We construct an outsourceable DHE scheme as shown in Figure 6. We define $\mathsf{DHE.MsgSp}(\mathsf{Mpk})$ and $\mathsf{DHE.FunSp}(\mathsf{Mpk})$ as arbitrary spaces that may parameterise our function family construction.

- $\mathsf{DHE.Setup}(1^\lambda)$
  1. $(\mathsf{Msk}, \mathsf{Mpk}) \leftarrow_\$ \mathsf{VFE.Setup}(1^\lambda)$
  2. Return $(\mathsf{Msk}, \mathsf{Mpk})$
- $\mathsf{DHE.Gen}(\mathsf{Mpk})$
  1. $(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{FHE.Gen}(1^\lambda)$
  2. Return $(\mathsf{sk}, (\mathsf{pk}, \mathsf{Mpk}))$
- $\mathsf{DHE.TKGen}(f, \mathsf{Msk})$
  1. $(\mathsf{TK}, \mathsf{h}_f) \leftarrow_\$ \mathsf{VFE.TKGen}(f^\star, \mathsf{Msk})$
  2. Return $(\mathsf{TK}, \mathsf{h}_f)$
- $\mathsf{DHE.Enc}(\mathsf{m}, (\mathsf{pk}, \mathsf{Mpk}))$
  1. $(\mathsf{c}, \mathsf{aux}) \leftarrow_\$ \mathsf{VFE.Enc}(\mathsf{m}, \mathsf{Mpk})$
  2. $\mathsf{c}_{\mathsf{hom}} \leftarrow_\$ \mathsf{FHE.Enc}(\mathsf{c}, \mathsf{pk})$
  3. Return $(\mathsf{c}_{\mathsf{hom}}, \mathsf{aux})$
- $\mathsf{DHE.Eval}(\mathsf{c}_{\mathsf{hom}}, \mathsf{TK}, (\mathsf{pk}, \mathsf{Mpk}))$
  1. $\mathsf{c} \leftarrow_\$ \mathsf{FHE.Eval}(\mathsf{c}_{\mathsf{hom}}, \mathsf{VFE.Dec}(\cdot, \mathsf{TK}), \mathsf{pk})$
  2. Return c
- $\mathsf{DHE.Dec}(\mathsf{c}, \mathsf{aux}, \mathsf{h}_f, \mathsf{sk}, \mathsf{Mpk})$
  1. $\mathsf{m} \leftarrow \mathsf{FHE.Dec}(\mathsf{c}, \mathsf{sk})$
  2. Parse $(y, \mathsf{t}) \leftarrow \mathsf{m}$
  3. If $\mathsf{VFE.Ver}((y, \mathsf{t}), \mathsf{h}_f, \mathsf{aux}, \mathsf{Mpk}) = \mathsf{F}$ Return $\perp$
  4. Return y

Fig. 6: A DHE scheme based on an FHE and a VFE scheme.

The correctness of the above scheme follows from the correctness of the underlying FHE and VFE schemes. The compactness of the construction follows from the compactness of the underlying FHE scheme, whereas its outsourceability follows from the outsourceability of the VFE. The security guarantees of the scheme are stated next. The precise statement and proof of the theorem is presented in Appendices F, G, and H.

Theorem 2 (Informal). *The DHE construction in Figure 6 provides input/output privacy, verifiability, and evaluation security if the VFE scheme is IND-CCA and verifiable, and the FHE is IND-CPA.*

7 Secure Verifiable Computation from DHE

Converting a DHE scheme to a VC scheme is straightforward. (For background on VC schemes we refer the reader to Appendix B.2.) Suppose DHE is a delegatable homomorphic encryption scheme. For any $\lambda \in \mathbb{N}$, and any $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{DHE.Setup}(1^\lambda)]$ set $\mathsf{VC.FunSp}(1^\lambda) := \mathsf{DHE.FunSp}(\mathsf{Mpk})$ and $\mathsf{VC.FunSp}(1^\lambda) := \mathsf{DHE.MsgSp}(\mathsf{Mpk})$ (we assume the spaces returned by DHE depend only on λ). Our construction is given in Figure 7.

- $\mathsf{VC.Gen}(f, 1^\lambda)$
  1. $(\mathsf{Msk}, \mathsf{Mpk}) \leftarrow_\$ \mathsf{DHE.Setup}(1^\lambda)$
  2. $(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{DHE.Gen}(\mathsf{Mpk})$
  3. $(\mathsf{TK}, \mathsf{h}_f) \leftarrow_\$ \mathsf{DHE.TKGen}(f, \mathsf{Msk})$
  4. Return $((\mathsf{h}_f, \mathsf{sk}, \mathsf{pk}), (\mathsf{TK}, \mathsf{pk}))$
- $\mathsf{VC.ProbGen}(\mathsf{m}, (\mathsf{h}_f, \mathsf{sk}, \mathsf{pk}))$
  1. $(\mathsf{c}, \mathsf{aux}) \leftarrow_\$ \mathsf{DHE.Enc}(\mathsf{m}, \mathsf{pk})$
  2. Return $(\mathsf{c}, \mathsf{aux})$
- $\mathsf{VC.Compute}(\mathsf{c}, (\mathsf{TK}, \mathsf{pk}))$
  1. $\mathsf{c}_{\mathsf{evl}} \leftarrow_\$ \mathsf{DHE.Eval}(\mathsf{c}, \mathsf{TK}, \mathsf{pk})$
  2. Return $\mathsf{c}_{\mathsf{evl}}$
- $\mathsf{VC.Verify}(\mathsf{c}_{\mathsf{evl}}, \mathsf{aux}, (\mathsf{h}_f, \mathsf{sk}, \mathsf{pk}))$
  1. Return $\mathsf{DHE.Dec}(\mathsf{c}_{\mathsf{evl}}, \mathsf{aux}, \mathsf{h}_f, \mathsf{sk})$

Fig. 7: A VC scheme based on a DHE scheme.

The correctness of the above scheme follows from the correctness of the underlying DHE scheme. Outsourceability of the VC scheme follows form the compactness and outsourceability of the DHE scheme. The security properties of this scheme are given next. The precise statements and proofs may be found in Appendices I and J.

Theorem 3 (Informal). *The VC scheme in Figure 7 is input/output private and* fully *verifiable if the underlying DHE scheme is input/output private and non-adaptively verifiable.*

It follows directly from this result that our DHE constructions implies the first VC scheme that preserves verifiability even if the adversary can adaptively observe verification results. Furthermore, our scheme preserves verifiability even if one removes the FHE layer. Hence, our results also answer an open question in [7]: it is possible to obtain a VC scheme without the cost of an FHE scheme, if one is willing to sacrifice I/O security.

REMARK. As for DHEs, we note that the VC scheme above does not provide input/output privacy in the presence of verification oracle. In scenarios where this level of confidentiality is required, a DHE scheme which provides CPA security in the presence of a verification oracle is needed. We leave this as a direction for future work.

8 Concluding Remarks

INSTANTIATING OUR DHE CONSTRUCTION. Currently, no fully secure PE scheme for general predicates exists, and hence our construction can still not be realised in its *full* generality. The recent scheme of De Caro et al. [6], however, does allow for the instantiation of our construction for a restricted class of functions. Concretely, the construction in [6] can be used to instantiate an FE scheme for the class $\mathcal{F}_k(\lambda)$ of all functions that can be implemented as the parallel composition of a polynomial number of k-CNF or k-DNF predicates, for a (globally) fixed k. Then, for any MAC tag-generation algorithm seen as a family of circuits MAC_λ indexed by the security parameter, the class of functions $\mathcal{F}^*_{k,\mathsf{MAC}}(\lambda)$ supported by the resulting DHE scheme is defined as

$$\mathcal{F}^*_{k,\mathsf{MAC}}(\lambda) := \{f \in \mathcal{F}_k(\lambda) \ : \ g_f \in \mathcal{F}_k(\lambda)\} \quad \text{where} \quad g_f(\mathsf{m},\mathsf{k}) := \mathsf{MAC}_\lambda((f(\mathsf{m}),\mathsf{h}_f),\mathsf{k}).$$

Note that one *cannot* argue that the construction in [6] leads to an FE scheme for all predicates as the usual Cook reduction from a CNF to a 3-CNF introduces new literals whose values depend on the input to the CNF. The input, contrary to an **NP**-completeness reduction, is not known in the functional encryption setting. This means that the values of the newly introduced variables cannot be set correctly. We, however, observe that if we fix the MAC scheme and a function f (or a set of functions) *a priori*, one can set up the DHE scheme to support g_f. In particular, in the strongly secure VC setting, which is one of our primary use-cases, the delegated function f is fixed at the set-up phase, and hence the generic construction can be instantiated using the De Caro et al. scheme.

EFFICIENCY CONSIDERATIONS. The complexities of the system setup and token-generation routines of our construction are the same as those in the underlying FE scheme. User key-generation and decryption routines are as efficient as those in the underlying FHE scheme, and hence relatively efficient (this is particularly important as the receiver is resource-constrained). The encryption algorithm homomorphically encrypts functional ciphertexts. Due to message expansion from the functional layer and bit-wise homomorphic encryption, we have a less efficient encryption/problem generation routine than that in [7]. Evaluation procedure requires homomorphically evaluating the decryption circuit of the functional encryption scheme. As the latter circuit involves operations such as computing pairings, and there are overheads associated with homomorphic evaluation, the outsourced computation in no longer $\Theta(f)$ (as required from an ideal scheme). We stress that our scheme is a conceptual solution to a complex problem, and the greater security guarantees and functionalities offered by our primitive, in our opinion, balance these efficiency losses. Furthermore, as it is a generic construction, efficiency will benefit from improvements in the underlying components.

OPEN PROBLEMS. The results in this paper solve some of the open problems in the area of Verifiable Computation. On the other hand, they open a number of avenues for further research.

An interesting direction is to uncover solutions that go beyond the non-adaptive and/or bounded token-extraction level of security that our DHE scheme achieves. Finding secure solutions under stronger models where the adversary maliciously generates keys, or auxiliary information used for verifiability is public, can also be considered. Simulation-based notions of security, given the impossibility result of Boneh et al. [4], deserve further investigation.

In terms of functionality, the proposed DHE primitive is somewhat limiting in what it offers: it is intrinsically 1-hop, and works only for functions of arity one; its delegation capabilities are not hierarchical; and its encryption routine is public and cannot offer function privacy for tokens. We leave such extensions of DHE for future work.

References

1. Mihir Bellare and Phillip Rogaway: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In *Advances in Cryptology – EUROCRYPT 2004*, LNCS 4004:409–426, Springer, 2006.
2. Siavosh Benabbas, Rosario Gennaro and Yevgeniy Vahlis: Verifiable Delegation of Computation over Large Datasets. In *Cryptology ePrint Archive*, Report 2011/132, 2011.
3. Dan Boneh and David Mandell Freeman: Homomorphic Signatures for Polynomial Functions. In *Advances in Cryptology – EUROCRYPT 2011*, LNCS 6632:149-168, Springer, 2011.
4. Dan Boneh, Amit Sahai, and Brent Waters: Functional Encryption: Definitions and Challenges. In *Cryptology ePrint Archive*, Report 2010/543, 2010.
5. Kai-Min Chung, Yael Kalai, and Salil P. Vadhan: Improved Delegation of Computation Using Fully Homomorphic Encryption. In *Advances in Cryptology – CRYPTO 2010*, LNCS 6223:483–501, Springer, 2010.
6. Angelo De Caro, Vincenzo Iovino, and Giuseppe Persiano: Efficient Fully Secure (Hierarchical) Predicate Encryption for Conjunctions, Disjunctions and k-CNF/DNF Formulae. In *Cryptology ePrint Archive*, Report 2010/492, 2010.
7. Rosario Gennaro, Craig Gentry, and Bryan Parno: Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In *Advances in Cryptology – CRYPTO 2010*, LNCS 6223:465–482, Springer, 2010.
8. Craig Gentry: Fully Homomorphic Encryption Using Ideal Lattices. In *41st ACM Symposium on Theory of Computing – STOC 2009*, pages 169-178, 2009.
9. Craig Gentry and Shai Halevi: A Working Implementation of Fully Homomorphic Encryption. Presentation at EUROCRYPT 2010 Rump Session, 2010.
10. Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan: i-Hop Homomorphic Encryption and Rerandomizable Yao Circuits. In *Advances in Cryptology – CRYPTO 2010*, LNCS 6223:155–172, Springer, 2010.
11. Jonathan Katz, Amit Sahai, and Brent Waters: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In *Advances in Cryptology – EUROCRYPT 2008*, LNCS 4965:146–162, Springer, 2008.
12. Hugo Krawczyk and Tal Rabin: Chameleon Signatures. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2000*, The Internet Society, 2000.
13. Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In *Advances in Cryptology – EUROCRYPT 2010*, LNCS 6110:62–91, Springer, 2010.
14. Yehuda Lindell and Benny Pinkas: A Proof of Security of Yao's Protocol for Two-Party Computation. In *J. Cryptology*, 22(2):161-188, 2009.
15. Adam O'Neill: Definitional Issues in Functional Encryption. In *Cryptology ePrint Archive*, Report 2010/556, 2010.
16. Tatsuaki Okamoto and Katsuyuki Takashima: Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In *Cryptology ePrint Archive*, Report 2010/563, 2010.
17. Emily Shen, Elaine Shi, and Brent Waters: Predicate Privacy in Encryption Systems. In *Theory of Cryptography – TCC 2009*, LNCS 5444:457–473, Springer, 2009.
18. Nigel P. Smart and Frederik Vercauteren: Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. In *Public Key Cryptography – PKC 2010*, LNCS 6056:420–443, Springer, 2010.
19. Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan: Fully Homomorphic Encryption over the Integers. In *Advances in Cryptology – EUROCRYPT 2010*, LNCS 6110:24–43, Springer, 2010.

A Chosen-Ciphertext Security for DHEs

The IND-CPA and verifiability security models both assume that the sender and the receiver are able to securely share aux without leaking it to the TA or to evaluators. It is interesting to analyse what is offered by a DHE scheme that is secure under both of them. To explore this, and also build confidence on the soundness of the models

introduced above, we now show that a secure DHE essentially achieves an intermediate level of non-malleability, as this is defined for public-key encryption schemes. Indeed, one can define an IND-CCA security level, in which the decryption oracle models an honest receiver (knowing the correct secret aux and a set of valid function fingerprints), that is satisfied by a DHE scheme that is both verifiable and ensures input/output privacy. This decryption oracle essentially always decrypts with respect to the fixed aux value, and rejects all ciphertexts encrypting messages that could have been trivially mauled from the challenge by a dishonest evaluator.

The IND-CCAx security of a DHE scheme for $\mathsf{x} \in \{1, 2\}$ requires the advantage of any PPT adversary $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\mathsf{ind\text{-}ccax}}_{\mathsf{DHE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathsf{IND\text{-}CCAx}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}\right] - 1$$

to be negligible, where game IND-CCAx is shown in Figure 8.

proc. Initialize(λ):	**proc. Token**(f):	**proc. Left-Right**$(\mathsf{m}_0, \mathsf{m}_1)$:
$b \leftarrow_\$ \{0,1\}$	$(\mathsf{TK}, \mathsf{h}_f) \leftarrow_\$ \mathsf{DHE.TKGen}(f, \mathsf{Msk})$	$(\mathsf{c}, \mathsf{aux}) \leftarrow_\$ \mathsf{DHE.Enc}(\mathsf{m}_b, \mathsf{pk})$
$\mathsf{TKList} \leftarrow \{\}$	$\mathsf{TKList} \leftarrow \mathsf{TKList} \cup \{(f, \mathsf{h}_f)\}$	Return c
$(\mathsf{Msk}, \mathsf{Mpk}) \leftarrow_\$ \mathsf{DHE.Setup}(1^\lambda)$	Return $(\mathsf{TK}, \mathsf{h}_f)$	
$(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{DHE.Gen}(\mathsf{Mpk})$		**proc. Finalize**(b'):
Return $(\mathsf{Mpk}, \mathsf{pk})$	**proc. Decrypt**$(\mathsf{c}, \mathsf{h}_f)$:	Return $(b = b')$
	$\mathsf{m} \leftarrow \mathsf{DHE.Dec}(\mathsf{c}, \mathsf{aux}, \mathsf{h}_f, \mathsf{sk})$	
	If $\mathsf{m} \in \{f(\mathsf{m}_b) : (f, \mathsf{h}_f) \in \mathsf{TKList}\}$ Return $\perp$	
	Return m	

Fig. 8: Game defining the IND-CCAx security of a DHE scheme. An adversary is legitimate if: 1) it calls **Left-Right** exactly once on two messages of equal length; 2) it never calls **Decrypt** on a h_f such that $(\star, \mathsf{h}_f) \notin \mathsf{TKList}$; and 3) if $\mathsf{x} = 1$ it does not call **Token** after calling **Left-Right**.

The following theorem establishes a relation between the models we have introduced. The intuition behind this result can also be stated as follows. An attacker against the IND-CCA game can follow one of two strategies. One strategy is to maul the challenge ciphertext to something different than that which is returned by the extracted tokens. This is essentially an attack against the TA, which is covered by the verifiability game. Otherwise, it may simply try to extract tokens that leak information about the challenge messages. But this attack is excluded by CPA security. It is also interesting to consider what happens if the decryption oracle accepts decryption queries on aux values other than the fixed one, e.g. if the receiver colludes with the dishonest evaluator. In the case of our concrete construction, the scheme becomes insecure.

Theorem 4 (VRF-CCAx $\wedge$ TA-IND-CPA $\Rightarrow$ IND-CCAx)**.** *Let* DHE *be a delegatable homomorphic encryption scheme. Let $\mathcal{A}$ be a* IND-CCAx *adversary against* DHE *placing at most $\mathbf{Q}^{\mathbf{Decrypt}}_{\mathsf{FE},\mathcal{A}}(\lambda)$ queries to its* **Decrypt** *procedure. Then there exists a* VRF-CCA2 *adversary $\mathcal{B}$, and a* TA-IND-CPA *adversary $\mathcal{C}$, both against the scheme* DHE*, such that for all $\lambda \in \mathbb{N}$ we have:*

$$\mathbf{Adv}^{\mathsf{ind\text{-}ccax}}_{\mathsf{DHE},\mathcal{A}}(\lambda) \leq \mathbf{Q}^{\mathbf{Decrypt}}_{\mathsf{DHE},\mathcal{A}}(\lambda) \cdot \mathbf{Adv}^{\mathsf{vrf\text{-}ccax}}_{\mathsf{DHE},\mathcal{B}}(\lambda) + \mathbf{Adv}^{\mathsf{ta\text{-}ind\text{-}cpa}}_{\mathsf{DHE},\mathcal{C}}(\lambda).$$

Proof. We prove this theorem via a sequence of games. Let $Q := \mathbf{Q}^{\mathbf{Decrypt}}_{\mathsf{DHE},\mathcal{A}}(\lambda)$. We define $\mathsf{Game}_{i^\star}$, for $i^\star = 0, \ldots, Q$, as follows. In $\mathsf{Game}_{i^\star}$ the j-th query to **Decrypt** is answered by $\perp$ if $j \leq i^\star$, when $j > i^\star$ we run the decryption procedure as in the IND-CCAx game. Note that Game_0 is identical to the IND-CCAx game whereas in Game_Q all **Decrypt** queries are answered by $\perp$. We first claim that any adversary $\mathcal{A}$ whose behaviour changes visibly between $\mathsf{Game}_{i^\star}$ and $\mathsf{Game}_{i^\star+1}$ can be converted into an attacker of verifiability $\mathcal{B}$ as shown in Figure 9.

We observe that $\mathcal{B}$'s simulation of both games is perfect until $\mathcal{A}$ performs the $(i^\star + 1)$-st query. This query will be rejected in $\mathsf{Game}_{i^\star+1}$ whereas in $\mathsf{Game}_{i^\star}$, adversary $\mathcal{A}$ will receive a correct decryption value. These two answers are different if the honest decryption is not $\perp$, and this query would constitute a valid forgery in the

verifiability game, as it would decrypt correctly under the $\mathsf{aux}^\star$ and h_f, but would produce something other than $f(\mathsf{m}_b)$ (according to the rules of IND-CCAx game). By the fundamental lemma of game hopping we have:

$$\Pr[\mathsf{Game}_{i^\star}{}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}] - \Pr[\mathsf{Game}_{i^\star+1}{}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}] \leq \Pr[\mathsf{VRF\text{-}CCAx}^{\mathcal{B}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}].$$

It follows that:

$$\Pr[\mathsf{IND\text{-}CCAx}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}] - \Pr[\mathsf{Game}_{Q}{}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}] \leq \mathbf{Q}^{\mathbf{Decrypt}}_{\mathsf{FE},\mathcal{A}}(\lambda) \cdot \Pr[\mathsf{VRF\text{-}CCAx}^{\mathcal{B}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}].$$

proc. $\mathbf{Initialize}(\lambda)$:	**proc.** $\mathbf{Decrypt}(\mathsf{c}, \mathsf{h}_f)$:	**proc.** $\mathbf{Left\text{-}Right}(\mathsf{m}_0, \mathsf{m}_1)$:
Get $(\mathsf{Mpk}, \mathsf{pk})$ from $\mathbf{Initialize}_{\mathsf{DHE}}(\lambda)$	$j \leftarrow j + 1$	Query $\mathbf{Encrypt}_{\mathsf{DHE}}(\mathsf{m}_b)$ to get c
$b \leftarrow \{0,1\}$; $j \leftarrow 0$; $X \leftarrow \epsilon$	If $j \leq i^\star$ Return $\perp$	Return c
Return $(\mathsf{Mpk}, \mathsf{pk})$	If $j = i^\star + 1$ Then	
	$X \leftarrow (\mathsf{c}, 1, \mathsf{h}_f)$	**proc.** $\mathbf{Finalize}(b')$:
proc. $\mathbf{Token}(f)$:	Return $\perp$	Call $\mathbf{Finalize}_{\mathsf{DHE}}(X)$
Query $\mathbf{Token}_{\mathsf{DHE}}(f)$ to get $(\mathsf{TK}, \mathsf{h}_f)$	If $j > i^\star + 1$ Then	
Return $(\mathsf{TK}, \mathsf{h}_f)$	Query $\mathbf{Decrypt}_{\mathsf{DHE}}(\mathsf{c}, 1, \mathsf{h}_f)$ to get m	
	Return m	

Fig. 9: Algorithm $\mathcal{B}$ against VRF-CCAx security of DHE, interpolating between games $i^\star$ and $i^\star + 1$.

To complete the proof, we note that Game_Q is a weaker version of the IND-CPA game for DHE where the adversary does not receive the master secret key, and is instead allowed to query a token extraction oracle. We therefore can define an attacker $\mathcal{C}$, shown in Figure 10 such that:

$$\Pr[\mathsf{Game}_{Q}{}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}] = \Pr[\mathsf{TA\text{-}IND\text{-}CPA}^{\mathcal{C}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}].$$

The theorem follows from the two previous claims.

proc. $\mathbf{Initialize}(\lambda)$:	**proc.** $\mathbf{Decrypt}_{\mathsf{DHE}}(\mathsf{c}, \mathsf{h}_f)$:	**proc.** $\mathbf{Left\text{-}Right}(\mathsf{m}_0, \mathsf{m}_1)$:
Get $(\mathsf{Msk}, \mathsf{Mpk}, \mathsf{pk})$ from $\mathbf{Initialize}_{\mathsf{DHE}}(\lambda)$	Return $\perp$	Query $\mathbf{Left\text{-}Right}_{\mathsf{DHE}}(\mathsf{m}_0, \mathsf{m}_1)$ to get c
Return $(\mathsf{Mpk}, \mathsf{pk})$		Return c
proc. $\mathbf{Token}(f)$:		**proc.** $\mathbf{Finalize}(b')$:
$(\mathsf{TK}, \mathsf{h}_f) \leftarrow \mathsf{DHE.TKGen}(f, \mathsf{Msk})$		Call $\mathbf{Finalize}_{\mathsf{DHE}}(b')$
Return $(\mathsf{TK}, \mathsf{h}_f)$		

Fig. 10: Algorithm $\mathcal{C}$ against TA-IND-CPA security of DHE.

□

B Background on FHE and VC, and a Formalisation of FE

B.1 Fully Homomorphic Public-Key Encryption

SYNTAX. A *homomorphic public-key encryption (HE) scheme* is specified by four PPT algorithms as follows.

1. $\mathsf{FHE.Gen}(1^\lambda)$: This is the key-generation algorithm and is run by the receiver. On input a security parameter, it outputs a receiver key-pair $(\mathsf{sk}, \mathsf{pk})$. We assume pk also contains a pair of compatible message and function spaces $\mathsf{FHE.MsgSp}(\mathsf{pk})$ and $\mathsf{FHE.FunSp}(\mathsf{pk})$.

2. $\mathsf{FHE.Enc}(\mathsf{m}, \mathsf{pk})$: This is the encryption algorithm and is run by the sender. On input a message m, and a receiver's public key pk, it returns a ciphertext c.
3. $\mathsf{FHE.Eval}(\mathsf{c}, f, \mathsf{pk})$: This is the evaluation algorithm, and is run by an evaluator. On input a ciphertext c, a circuit f, and a receiver's public key pk, it outputs a ciphertext $\mathsf{c}_{\mathsf{evl}}$.
4. $\mathsf{FHE.Dec}(\mathsf{c}, \mathsf{sk})$: This is the deterministic decryption algorithm and is run by the receiver. On input a ciphertext c, a secret key sk, it returns either a message m or a special failure symbol $\perp$.

CORRECTNESS. An HE scheme is correct if for any $\lambda \in \mathbb{N}$, any $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{FHE.Gen}(1^\lambda)]$, any $\mathsf{m} \in \mathsf{FHE.MsgSp}(\mathsf{pk})$, any $\mathsf{c} \in [\mathsf{FHE.Enc}(\mathsf{m}, \mathsf{pk})]$, any function $f \in \mathsf{FHE.FunSp}(\mathsf{pk})$, and any $\mathsf{c}_{\mathsf{evl}} \in [\mathsf{FHE.Eval}(\mathsf{c}, f, \mathsf{pk})]$, we have that $\mathsf{FHE.Dec}(\mathsf{c}_{\mathsf{evl}}, \mathsf{sk}) = f(\mathsf{m})$. We do not necessarily require correctness over freshly created ciphertexts.

PLAINTEXT SECURITY. The IND-CPA security of an HE scheme requires the advantage of any PPT adversary $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{FHE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathsf{IND\text{-}CPA}^{\mathcal{A}}_{\mathsf{FHE}}(\lambda) \Rightarrow \mathsf{T}\right] - 1,$$

to be negligible, where game IND-CPA is shown in Figure 11.

proc. Initialize(λ):	**proc. Left-Right**$(\mathsf{m}_0, \mathsf{m}_1)$:	**proc. Finalize**(b'):
$b \leftarrow_{\$} \{0,1\}$	$\mathsf{c} \leftarrow_{\$} \mathsf{FHE.Enc}(\mathsf{m}_b, \mathsf{pk})$	Return $(b = b')$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow_{\$} \mathsf{FHE.Gen}(1^\lambda)$	Return c	
Return pk		

Fig. 11: Game defining the IND-CPA security of an HE scheme. An adversary is legitimate if it calls **Left-Right** exactly once on two messages of equal length.

COMPACTNESS. A homomorphic encryption scheme is compact if there exists a fixed polynomial bound $\mathrm{B}(\cdot)$ so that for any key-pair $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{FHE.Gen}(1^\lambda)]$, any function $f \in \mathsf{FHE.FunSp}(\mathsf{pk})$, and any message $\mathsf{m} \in \mathsf{FHE.MsgSp}(\mathsf{pk})$, any ciphertext $\mathsf{c} \in [\mathsf{FHE.Enc}(\mathsf{m}, \mathsf{pk})]$, and any evaluated ciphertext $\mathsf{c}_{\mathsf{evl}} \in [\mathsf{FHE.Eval}(\mathsf{c}, f, \mathsf{pk})]$, the size of $\mathsf{c}_{\mathsf{evl}}$ is at most $\mathrm{B}(\lambda + |f(\mathsf{m})|)$ (independently of the size of f).

FULLY HOMOMORPHIC ENCRYPTION. A homomorphic encryption scheme is called fully homomorphic if for any $\lambda \in \mathbb{N}$, the function space $\mathsf{FHE.FunSp}(\mathsf{pk})$ contains all functions on the message space $\mathsf{FHE.MsgSp}(\mathsf{pk})$, and the latter contains at least two elements.

B.2 Verifiable Computation

In this section we present the notion of an outsourceable and verifiable computation scheme as in [7].

SYNTAX. A *verifiable computation (VC) scheme* for a pair of compatible spaces $\mathsf{VC.MsgSp}(1^\lambda)$ and $\mathsf{VC.FunSp}(1^\lambda)$ is specified by four PPT algorithms as follows.

1. $\mathsf{VC.Gen}(f, 1^\lambda)$: This is the key-generation algorithm and is run by the delegator. On input a function f and a security parameter 1^λ, this algorithm returns a key pair $(\mathsf{sk}, \mathsf{pk})$
2. $\mathsf{VC.ProbGen}(\mathsf{m}, \mathsf{sk})$: This is the problem generation algorithm and is run by the delegator. On input a message m and a secret key sk, this returns a ciphertext c and some auxiliary information aux.
3. $\mathsf{VC.Compute}(\mathsf{c}, \mathsf{pk})$: This is the evaluation algorithm and is run by the worker. On input a ciphertext c and a public key pk, this algorithm outputs a new ciphertext $\mathsf{c}_{\mathsf{evl}}$.
4. $\mathsf{VC.Verify}(\mathsf{c}_{\mathsf{evl}}, \mathsf{aux}, \mathsf{sk})$: This is the deterministic decryption algorithm and is run by the delegator. On input a $\mathsf{c}_{\mathsf{evl}}$ this algorithm outputs a message m or a special failure symbol $\perp$.

CORRECTNESS. A VC scheme is correct if for any $\lambda \in \mathbb{N}$, any $f \in \mathsf{VC.FunSp}(1^\lambda)$, any $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{VC.Gen}(f, 1^\lambda)]$, any $\mathsf{m} \in \mathsf{VC.MsgSp}(1^\lambda)$, any $(\mathsf{c}, \mathsf{aux}) \in [\mathsf{VC.ProbGen}(\mathsf{m}, \mathsf{sk})]$, and any $\mathsf{c_{evl}} \in [\mathsf{VC.Compute}(\mathsf{c}, \mathsf{pk})]$, we have that $\mathsf{VC.Verify}(\mathsf{c_{evl}}, \mathsf{aux}, \mathsf{sk}) = f(\mathsf{m})$.

OUTSOURCEABILITY. A VC scheme is outsourceable if:

1. For any $c > 0$ and sufficiently large λ, we have that for any $f \in \mathsf{VC.FunSp}(1^\lambda)$, any $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{VC.Gen}(f, 1^\lambda)]$, any $\mathsf{m} \in \mathsf{VC.MsgSp}(1^\lambda)$, the time-complexity of $\mathsf{VC.ProbGen}(\mathsf{m}, \mathsf{sk})$ is at most c times the time-complexity of $f(\mathsf{m})$; and
2. There is a fixed polynomial $\mathrm{B}(\cdot)$ such that for any $f \in \mathsf{VC.FunSp}(1^\lambda)$, any $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{VC.Gen}(f, 1^\lambda)]$, any $\mathsf{m} \in \mathsf{VC.MsgSp}(1^\lambda)$, and any $\mathsf{c_{evl}} \in [\mathsf{VC.Compute}(\mathsf{c}, \mathsf{pk})]$, the sizes of $(\mathsf{m}, \mathsf{sk})$ and $(\mathsf{c_{evl}}, \mathsf{aux}, \mathsf{sk})$ are at most $\mathrm{B}(\lambda + |f(\mathsf{m})|)$ (independently of the size of f).

REMARK. The definition of outsourceability adapted in [7] requires the total time-complexity of $\mathsf{VC.Verify}(\mathsf{c_{evl}}, \mathsf{aux}, \mathsf{sk})$ to be also $o(T)$ where T is the time-complexity of $f(\mathsf{m})$. The proposed scheme in [7] as well as our scheme achieve the slightly different notion of outsourceability defined above.

If we denote a bound on the time-complexity of VC.Verify by $\mathrm{B}'(\cdot)$, our outsourceability notion implies that in [7] as long as the time-complexity of f is at least $c \cdot \mathrm{B}'(\mathrm{B}(\lambda + |f(\mathsf{m})|))$ for any $c > 0$ and sufficiently large λ. Note also the converse implication for general f does not hold as the time-complexity of VC.Verify could be $o(T)$ without having to be bounded by a fixed polynomial B.

INPUT/OUTPUT SECURITY. The IND-CPA security of a VC scheme requires that for any $f \in \mathsf{FunSp}(1^\lambda)$, the advantage of any PPT adversary $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{f,\mathsf{VC},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathsf{IND\text{-}CPA}^{\mathcal{A}}_{f,\mathsf{VC}}(\lambda) \Rightarrow \mathsf{T}\right] - 1,$$

to be negligible, where game IND-CPA is shown in Figure 12.

proc. Initialize(f, λ):	**proc. PubProbGen**(m):	**proc. Left-Right**$(\mathsf{m}_0, \mathsf{m}_1)$:	**proc. Finalize**(b'):
$b \leftarrow_\$ \{0,1\}$	$(\mathsf{c}, \mathsf{aux}) \leftarrow_\$ \mathsf{VC.ProbGen}(\mathsf{m}, \mathsf{sk})$	$\mathsf{c} \leftarrow_\$ \mathsf{VC.ProbGen}(\mathsf{m}_b, \mathsf{sk})$	Return $(b = b')$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{VC.Gen}(f, 1^\lambda)$	Return c	Return c	
Return pk			

Fig. 12: Game defining the IND-CPA security of a VC scheme. An adversary is legitimate if it calls **Left-Right** exactly once and on two messages of equal length.

VERIFIABILITY. The VRF-CCAx security of a VC scheme for $\mathsf{x} \in \{0,1\}$ requires that for any $f \in \mathsf{FunSp}(1^\lambda)$, the advantage of any PPT adversary $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\mathsf{vrf\text{-}ccax}}_{f,\mathsf{VC},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathsf{VRF\text{-}CCAx}^{\mathcal{A}}_{f,\mathsf{VC}}(\lambda) \Rightarrow \mathsf{T}\right] - 1,$$

to be negligible, where game VRF-CCAx is shown in Figure 13.

B.3 Functional Encryption

SYNTAX. A *functional encryption (FE) scheme* is specified by four PPT algorithms as follows.

1. $\mathsf{FE.Setup}(1^\lambda)$: This is the setup algorithm. On input a security parameter 1^λ, it outputs a master public key Mpk and a master secret key Msk. We assume Mpk also contains a pair of compatible message and function spaces $\mathsf{FE.MsgSp}(\mathsf{Mpk})$ and $\mathsf{FE.FunSp}(\mathsf{Mpk})$.

proc. Initialize(f, λ):	**proc. PubProbGen**(m):	**proc. PubVerify**(c, i):	**proc. Finalize**$(\mathsf{c}^\star, i)$:
$\mathsf{List} \leftarrow \{\}; i \leftarrow 0$	$(\mathsf{c}, \mathsf{aux}) \leftarrow_\$ \mathsf{VC.ProbGen}(\mathsf{m}, \mathsf{sk})$	Let $(\mathsf{m}, \mathsf{aux})$ be s.t.	If $(i, \star, \star) \notin \mathsf{List}$ Return F
$(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{VC.Gen}(f, 1^\lambda)$	$i \leftarrow i + 1$	$(i, \mathsf{m}, \mathsf{aux}) \in \mathsf{List}$	Let $(\mathsf{m}, \mathsf{aux})$ be s.t. $(i, \mathsf{m}, \mathsf{aux}) \in \mathsf{List}$
Return pk	$\mathsf{List} \leftarrow \mathsf{List} \cup \{(i, \mathsf{m}, \mathsf{aux})\}$	$\mathsf{m} \leftarrow \mathsf{VC.Verify}(\mathsf{c}, \mathsf{aux}, \mathsf{sk})$	$\mathsf{m}^\star \leftarrow \mathsf{VC.Verify}(\mathsf{c}^\star, \mathsf{aux}, \mathsf{sk})$
	Return c	Return m	If $\mathsf{m}^\star = \perp$ Return F
			Return $(\mathsf{m}^\star \neq f(\mathsf{m}))$

Fig. 13: Game defining the VRF-CCAx security of a VC scheme. An adversary is legitimate if: 1) it never calls $\mathbf{PubVerify}$ with an i such that $(i, \star, \star) \notin \mathsf{List}$; and 2) it does not call $\mathbf{PubVerify}$ when $\mathsf{x} = 0$.

2. $\mathsf{FE.TKGen}(f, \mathsf{Msk})$: This is the token generation algorithm. On input a function f, and a master secret key Msk, it outputs a token TK for f.
3. $\mathsf{FE.Enc}(\mathsf{m}, \mathsf{Mpk})$: This is the encryption algorithm. On input a message m, it outputs a ciphertext c.
4. $\mathsf{FE.Dec}(\mathsf{c}, \mathsf{TK})$: This is the deterministic decryption algorithm. On input a ciphertext c, and a token TK, it outputs a message m or a special failure symbol $\perp$.

CORRECTNESS. An FE scheme is correct if for any $\lambda \in \mathbb{N}$, any $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{FE.Setup}(1^\lambda)]$, any $\mathsf{m} \in \mathsf{FE.MsgSp}(\mathsf{Mpk})$, any $\mathsf{c} \in [\mathsf{FE.Enc}(\mathsf{m}, \mathsf{Mpk})]$, any $f \in \mathsf{FE.FunSp}(\mathsf{Mpk})$, and any $\mathsf{TK} \in [\mathsf{FE.TKGen}(f, \mathsf{Msk})]$, we have that $\mathsf{FE.Dec}(\mathsf{c}, \mathsf{TK}) = f(\mathsf{m})$.

OUTSOURCEABILITY. An FE scheme is outsourceable if for every $c > 0$ and sufficiently large λ we have that for any $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{FE.Setup}(1^\lambda)]$, any message $\mathsf{m} \in \mathsf{FE.MsgSp}(\mathsf{Mpk})$, and any $f \in \mathsf{FE.FunSp}(\mathsf{Mpk})$, the time-complexity of $\mathsf{FE.Enc}(\mathsf{m}, \mathsf{Mpk})$ is at most c times the time-complexity of $f(\mathsf{m})$. Put differently, for every f, the time-complexity of $\mathsf{FE.Enc}(\cdot, \cdot)$ is $o(|f|)$, where $|f|$ denotes the number of gates that f contains (the circuit size of f).

REMARK. Given a function space FunSp containing polynomially many functions, one can trivially construct a functional encryption scheme supporting FunSp from a standard public-key encryption scheme as follows. A fresh secret/public key-pair is assigned to each function. To encrypt m one encrypts $f(\mathsf{m})$ for all $f \in \mathsf{FunSp}$ under f's public key. Decryption is performed using f's secret key. Note, however, that this construction (which precomputes f values locally during encryption) does not satisfy our outsourceability criterion.

FULLY FUNCTIONAL ENCRYPTION. A functional encryption scheme FE is called fully functional if, for any $\lambda \in \mathbb{N}$, the function space $\mathsf{FE.FunSp}(\mathsf{Mpk})$ contains all functions on the message space $\mathsf{FE.MsgSp}(\mathsf{Mpk})$, and the latter has at least two elements.

PLAINTEXT SECURITY. The IND-CCAx security of an FE scheme for $\mathsf{x} \in \{0, 1, 2\}$ requires the advantage of any PPT adversary $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\mathsf{ind\text{-}ccax}}_{\mathsf{FE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathsf{IND\text{-}CCAx}^{\mathcal{A}}_{\mathsf{FE}}(\lambda) \Rightarrow \mathsf{T}\right] - 1,$$

to be negligible, where game IND-CCAx is shown in Figure 14.

A fully functional encryption scheme can be constructed from a predicate-only PE scheme that works for all predicates in the following way. The global set-up algorithm is the same as that of the PE scheme. Token extraction, when queried on a function mapping m-bit to n-bit strings, first defines n predicates, where predicate i corresponds to the i-th output bit of f. It then uses the PE token-generation algorithm to obtain tokens for all n predicates, and returns them. Decryption simply recovers the result of computing f bit by bit by PE-decrypting the ciphertext with all the secret keys provided in the token. If the PE scheme is adaptively attribute hiding under CCA attacks, the CCA security of the resulting FE scheme would also follow from a simple reduction.

We now formalise this folklore construction of a functional encryption scheme from a restricted FE scheme for $0/1$-functions. Predicate-only encryption schemes can be seen as an example of such a restricted FE scheme.

proc. Initialize(λ):
$b \leftarrow_{\$} \{0,1\}$; $\mathsf{List} \leftarrow \{\}$; $\mathsf{TKList} \leftarrow \{\}$
$(\mathsf{Msk}, \mathsf{Mpk}) \leftarrow_{\$} \mathsf{FE.Setup}(1^\lambda)$
Return Mpk

proc. Token(f):
$\mathsf{TK} \leftarrow_{\$} \mathsf{FE.TKGen}(f, \mathsf{Msk})$
$\mathsf{TKList} \leftarrow \mathsf{TKList} \cup \{f\}$
Return TK

proc. Decrypt(c, f):
$\mathsf{TK} \leftarrow_{\$} \mathsf{FE.TKGen}(f, \mathsf{Msk})$
$\mathsf{m} \leftarrow \mathsf{FE.Dec}(\mathsf{c}, \mathsf{TK})$
Return m

proc. Left-Right$(\mathsf{m}_0, \mathsf{m}_1)$:
$\mathsf{c} \leftarrow_{\$} \mathsf{FE.Enc}(\mathsf{m}_b, \mathsf{Mpk})$
$\mathsf{List} \leftarrow \mathsf{List} \cup \{\mathsf{c}\}$
Return c

proc. Finalize(b'):
Return $(b = b')$

Fig. 14: Game defining the IND-CCAx security of an FE scheme. An adversary is legitimate if: 1) it calls **Left-Right** exactly once on two messages of equal length; 2) it never calls **Decrypt** with a $\mathsf{c} \in \mathsf{List}$; 3) It never calls **Token** with an f such that $f(\mathsf{m}_0) \neq f(\mathsf{m}_1)$; 4) if $\mathsf{x} = 0$ it does not call **Decrypt**; and 5) if $\mathsf{x} = 1$ it does not call **Decrypt** after calling **Left-Right**.

- $\mathsf{FE.Setup}(1^\lambda)$
  1. $(\mathsf{Msk}, \mathsf{Mpk}) \leftarrow_{\$} \mathsf{PE.Setup}(1^\lambda)$
  2. Return $(\mathsf{Msk}, \mathsf{Mpk})$
- $\mathsf{FE.TKGen}(f, \mathsf{Msk})$
  1. For $i = 1 \ldots, n$
     Define $f_i := i$-th bit of f
  2. $\mathsf{TK}_i \leftarrow \mathsf{PE.TKGen}(f_i, \mathsf{Msk})$
  3. Return $(\mathsf{TK}_1, \ldots, \mathsf{TK}_n)$
- $\mathsf{FE.Enc}(\mathsf{m}, \mathsf{Mpk})$
  1. $\mathsf{c} \leftarrow_{\$} \mathsf{PE.Enc}(\mathsf{m}, \mathsf{Mpk})$
  2. Return c
- $\mathsf{FE.Dec}(\mathsf{c}, (\mathsf{TK}_1, \ldots, \mathsf{TK}_n))$
  1. For $i = 1 \ldots, n$
     $\mathsf{m}_i \leftarrow \mathsf{PE.Dec}(\mathsf{c}, \mathsf{TK}_i)$
  2. Return $(\mathsf{m}_1, \ldots, \mathsf{m}_n)$

Fig. 15: An FE scheme based on a PE scheme.

It is easy to check that the FE scheme in Figure 15 is correct if the underlying PE scheme is correct. The security of the construction is also straightforward to establish.

Theorem 5. *Let $\mathcal{A}$ be a PPT adversary against the* IND-CCAx *security of the FE scheme in Figure 15. Then there exists a PPT adversary $\mathcal{B}$ against the* IND-CCAx *security of the underlying PE scheme such that for all $\lambda \in \mathbb{N}$ we have:*

$$\mathbf{Adv}^{\mathsf{ind\text{-}ccax}}_{\mathsf{FE},\mathcal{A}}(\lambda) = \mathbf{Adv}^{\mathsf{ind\text{-}ccax}}_{\mathsf{PE},\mathcal{B}}(\lambda).$$

Proof. Let $\mathcal{A}$ be an algorithm as in the statement of the theorem. We construct the algorithm $\mathcal{B}$ against the IND-CCAx security of PE as shown in Figure 16.

proc. Initialize(λ):
Get Mpk from $\mathbf{Initialize}_{\mathsf{PE}}(\lambda)$
Return Mpk

proc. Token(f):
For $i = 1 \ldots, n$
  Define $f_i := i$-th bit of f
  Query $\mathbf{Token}(f_i)$ to get TK_i
Return $(\mathsf{TK}_1, \ldots, \mathsf{TK}_n)$

proc. Left-Right$(\mathsf{m}_0, \mathsf{m}_1)$:
Query $\mathbf{Left\text{-}Right}_{\mathsf{PE}}(\mathsf{m}_0, \mathsf{m}_1)$ to get c
Return c

proc. Decrypt(c, f):
For $i = 1 \ldots, n$
  Define $f_i := i$-th bit of f
  Query $\mathbf{Decrypt}_{\mathsf{PE}}(\mathsf{c}, f_i)$ to get m_i
Return $(\mathsf{m}_1, \ldots, \mathsf{m}_n)$

proc. Finalize(b'):
Call $\mathbf{Finalize}_{\mathsf{PE}}(b')$

Fig. 16: Algorithm $\mathcal{B}$ against the IND-CCAx security of PE.

We note that $\mathcal{B}$ wins its game exactly when $\mathcal{A}$ wins its game. This follows from two observations as follows. First, if $\mathcal{A}$ is legitimate, then so is $\mathcal{B}$: if $\mathcal{A}$ always queries its **Token** oracle with functions f such that $f(\mathsf{m}_0) = f(\mathsf{m}_1)$, then it is necessarily the case that all the bits are also equal, i.e. $f_i(\mathsf{m}_0) = f_i(\mathsf{m}_1)$.

Second, from the description of $\mathcal{B}$ it is clear that:

$$\Pr\left[\mathsf{IND\text{-}CCAx}^{\mathcal{A}}_{\mathsf{FE}}(\lambda) \Rightarrow \mathsf{T} | b = 1\right] = \Pr\left[\mathsf{IND\text{-}CCAx}^{\mathcal{B}}_{\mathsf{PE}}(\lambda) \Rightarrow \mathsf{T} | b = 1\right],$$

and

$$\Pr\left[\mathsf{IND\text{-}CCAx}^{\mathcal{A}}_{\mathsf{FE}}(\lambda) \Rightarrow \mathsf{T} | b = 0\right] = \Pr\left[\mathsf{IND\text{-}CCAx}^{\mathcal{B}}_{\mathsf{PE}}(\lambda) \Rightarrow \mathsf{T} | b = 0\right].$$

The theorem follows. □

C MACs with Chameleon Keys

We now introduce a new type of message authentication code (MAC), reminiscent of chameleon hashing [12].

SYNTAX. An n-*key-chameleon MAC scheme* is specified by three PPT algorithms as follows.

1. $\mathsf{MAC.Setup}(1^\lambda)$: This is the setup algorithm. On input the security parameter, this algorithm return a pair $(\mathsf{td}, \mathsf{mk})$ consisting of a trapdoor td and public parameters mk. We assume mk also contains a message space $\mathsf{MAC.MsgSp}(\mathsf{mk})$ and a key space $\mathsf{MAC.KeySp}(\mathsf{mk})$.
2. $\mathsf{MAC.Tag}(\mathsf{m}, \mathsf{k})$: This is the deterministic tagging algorithm. On input the global parameters mk, a message m, and a secret key k, this algorithm returns a tag t.
3. $\mathsf{MAC.Col}(\mathsf{td}, \mathsf{m}_1, \ldots, \mathsf{m}_n, \mathsf{k})$: This is the collision-finding algorithm. On input a trapdoor td, n messages, and a secret key k this algorithm returns a new secret key k'.

CORRECTNESS. An n-key-chameleon MAC scheme is correct if for any $\lambda \in \mathbb{N}$, any $(\mathsf{td}, \mathsf{mk}) \in [\mathsf{MAC.Setup}(1^\lambda)]$, any $\mathsf{m}_i \in \mathsf{MAC.MsgSp}(\mathsf{mk})$ for $i = 1, \ldots, n$, any $\mathsf{k} \in \mathsf{MAC.KeySp}(\mathsf{mk})$, and any $\mathsf{k}' \in [\mathsf{MAC.Col}(\mathsf{td}, \mathsf{m}_1, \ldots, \mathsf{m}_n, \mathsf{k})]$ we have that $\mathsf{MAC.Tag}(\mathsf{m}_i, \mathsf{k}, \mathsf{mk}) = \mathsf{MAC.Tag}(\mathsf{m}_i, \mathsf{k}', \mathsf{mk})$ for all $i = 1, \ldots, n$.

UNFORGEABILITY. The UF-CMA security of an n-key-chameleon MAC requires the advantage of any PPT adversary $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\mathsf{uf\text{-}cma}}_{\mathsf{MAC},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{MAC}}(\lambda) \Rightarrow \mathsf{T}\right],$$

to be negligible, where game UF-CMA is shown in Figure 17. Note that this is essentially a slightly stronger version of a non-adaptive chosen message attack, as k' allows the adversary to compute a tag on the chosen messages.

proc. Initialize(λ):	**proc. Collision**$(\mathsf{m}_1, \ldots, \mathsf{m}_n)$:	**proc. Finalize**(m, t):
$(\mathsf{td}, \mathsf{mk}) \leftarrow_\$ \mathsf{MAC.Setup}(1^\lambda)$	$\mathsf{k}' \leftarrow \mathsf{MAC.Col}(\mathsf{td}, \mathsf{m}_1, \ldots, \mathsf{m}_n, \mathsf{k})$	If $\mathsf{m} \in \{\mathsf{m}_1, \ldots, \mathsf{m}_n\}$ Return F
$\mathsf{k} \leftarrow_\$ \mathsf{MAC.KeySp}(\mathsf{mk})$	Return k'	Return $(\mathsf{MAC.Tag}(\mathsf{m}, \mathsf{k}, \mathsf{mk}) = \mathsf{t})$
Return mk		

Fig. 17: Game defining the UF-CMA security of an n-key chameleon MAC. An adversary is legitimate if it calls **Collision** exactly once.

REMARK. Note that adversary gets to see a key that collides with the challenge key on all tags that it sees, but still this should not help her forge a new tag.

BUILDING AN n-KEY CHAMELEON MAC. One can build an n-key-chameleon MAC from the n-time information-theoretically secure MAC with tagging algorithm $\mathsf{MAC.Tag}(\mathsf{m}, (a_n, \ldots, a_0), \mathsf{mk}) := \sum_{i=0}^{n} a_i \mathsf{m}^i$ with $a_i, \mathsf{m} \in \mathbb{Z}_p$ for a prime p. There is no need for global parameters (except for the appropriate message and key space definitions) and no need for a trapdoor. Key generation returns a random $n+1$ tuple $(a_n, \ldots, a_0)$. To find a colliding key $\mathsf{k}' = (a'_n, \ldots, a'_0)$ given n messages, as required by the correctness condition, one solves the system of n equations in $n+1$ unknowns. Since this system is under-defined, the new computed key can be made to look completely random. Hence, providing this key to the adversary leaks no information. A forgery as required by the game above thus translates to an n-time forgery, which is infeasible in the information-theoretical sense.

D Indistinguishability of the VFE Construction

Theorem 6 (Indistinguishability). *Let $\mathcal{A}$ be a PPT adversary against the* IND-CCAx *security of the VFE scheme in Figure 5. Then there exists a PPT adversary $\mathcal{B}$ against* IND-CCAx *security of the FE scheme that such that for all $\lambda \in \mathbb{N}$ we have:*

$$\mathbf{Adv}^{\mathsf{ind\text{-}ccax}}_{\mathsf{VFE},\mathcal{A}}(\lambda) = \mathbf{Adv}^{\mathsf{ind\text{-}ccax}}_{\mathsf{FE},\mathcal{B}}(\lambda).$$

Proof. Let $\mathcal{A}$ be an algorithm as in the statement of the theorem. We construct an algorithm $\mathcal{B}$ against the IND-CCAx security of the underlying functional encryption scheme as shown in Figure 18.

proc. $\mathbf{Initialize}(\lambda)$:
Get Mpk from $\mathbf{Initialize}_{\mathsf{FE}}(\lambda)$
$(\mathsf{td}, \mathsf{mk}) \leftarrow_{\$} \mathsf{MAC.Setup}(1^{\lambda})$
$\mathsf{hk} \leftarrow_{\$} \mathsf{CRH.Gen}(1^{\lambda})$
Return $(\mathsf{Mpk}, \mathsf{mk}, \mathsf{hk})$

proc. $\mathbf{Token}(f)$:
$\mathsf{h}_f \leftarrow \mathsf{CRH.H}(\langle f \rangle, \mathsf{hk})$
$f^{\star}(x_1, x_2) := (f(x_1), \mathsf{MAC.Tag}((f(x_1), \mathsf{h}_f), x_2, \mathsf{Mpk}))$
Query $\mathbf{Token}_{\mathsf{FE}}(f^{\star})$ to get TK
Return $(\mathsf{TK}, \mathsf{h}_f)$

proc. $\mathbf{Decrypt}(\mathsf{c}, f)$:
$\mathsf{h}_f \leftarrow \mathsf{CRH.H}(\langle f \rangle, \mathsf{hk})$
$f^{\star}(x_1, x_2) := (f(x_1), \mathsf{MAC.Tag}((f(x_1), \mathsf{h}_f), x_2, \mathsf{Mpk}))$
Query $\mathbf{Decrypt}_{\mathsf{FE}}(\mathsf{c}, f^{\star})$ to get m
Parse $(y, \mathsf{t}) \leftarrow \mathsf{m}$
Return (y, t)

proc. $\mathbf{Left\text{-}Right}(\mathsf{m}_0, \mathsf{m}_1)$:
$\mathsf{k} \leftarrow_{\$} \mathsf{MAC.KeySp}(\mathsf{mk})$
Query $\mathbf{Left\text{-}Right}_{\mathsf{FE}}((\mathsf{m}_0, \mathsf{k}), (\mathsf{m}_1, \mathsf{k}))$ to get c
Return (c, k)

proc. $\mathbf{Finalize}(b')$:
Call $\mathbf{Finalize}_{\mathsf{FE}}(b')$

Fig. 18: Algorithm $\mathcal{B}$ against the IND-CCAx security of FE.

From the description of $\mathcal{B}$ it is clear that, if $\mathcal{A}$ is a legitimate adversary then so is $\mathcal{B}$, and furthermore:

$$\Pr\left[\mathsf{IND\text{-}CCAx}^{\mathcal{A}}_{\mathsf{VFE}}(\lambda) \Rightarrow \mathsf{T} | b = 1\right] = \Pr\left[\mathsf{IND\text{-}CCAx}^{\mathcal{B}}_{\mathsf{FE}}(\lambda) \Rightarrow \mathsf{T} | b = 1\right],$$

and

$$\Pr\left[\mathsf{IND\text{-}CCAx}^{\mathcal{A}}_{\mathsf{VFE}}(\lambda) \Rightarrow \mathsf{T} | b = 0\right] = \Pr\left[\mathsf{IND\text{-}CCAx}^{\mathcal{B}}_{\mathsf{FE}}(\lambda) \Rightarrow \mathsf{T} | b = 0\right].$$

The theorem follows. □

E Verifiability of the VFE Construction

Theorem 7 (Verifiability). *Let $\mathcal{A}$ be a PPT adversary against the* VRF-CCA1 *security of the VFE scheme in Figure 5 placing at most $\mathbf{Q}^{\mathbf{Encrypt}}_{\mathsf{VFE},\mathcal{A}}(\lambda)$ queries to its $\mathbf{Encrypt}$ procedure, and at most n (non-adaptive) queries to its $\mathbf{Token}$ procedure. Then there exists a PPT adversary $\mathcal{B}$ against the* IND-CCA1 *security of the FE scheme and a PPT adversary $\mathcal{C}$ against the* UF-CMA *security of the n-key-chameleon MAC such that for all $\lambda \in \mathbb{N}$ we have:*

$$\mathbf{Adv}^{\mathsf{vrf\text{-}cca1}}_{\mathsf{VFE},\mathcal{A}}(\lambda) \leq \mathbf{Q}^{\mathbf{Encrypt}}_{\mathsf{VFE},\mathcal{A}}(\lambda) \cdot (\mathbf{Adv}^{\mathsf{ind\text{-}cca1}}_{\mathsf{FE},\mathcal{B}}(\lambda) + \mathbf{Adv}^{\mathsf{uf\text{-}cma}}_{\mathsf{MAC},\mathcal{C}}(\lambda)).$$

Proof. Recall that we are assuming a non-adaptive token attack where the adversary must make all his token extraction queries before calling encryption.

The proof of this theorem proceeds through a sequence of games as follows. We let Game'_0 to be identical to the VRF-CCA1 security game for VFE. We modify Game'_0 to Game_0 by introducing a simple change. We select a random encryption query $i^{\star}$ and allow the adversary to win only when attacking the associated ciphertext. We note that $i^{\star}$ is completely hidden from $\mathcal{A}$'s view in Game'_0. Furthermore, observing that the probability that $i = i^{\star}$ is $1/\mathbf{Q}^{\mathbf{Encrypt}}_{\mathsf{VFE},\mathcal{A}}(\lambda)$, accounts for the multiplicative factor in the Theorem statement.

For Game_1 we introduce an additional change in the encryption oracle. For query $i^{\star}$, a key k' is generated through

proc. Initialize(λ):
$i^\star \leftarrow_\$ \{1, \ldots, Q\}$
Other steps as in VRF-CCA1

proc. Token(f):
As in VRF-CCA1

proc. Decrypt(c, f):
As in VRF-CCA1

proc. Encrypt(m):
$\mathsf{k} \leftarrow_\$ \mathsf{MAC.KeySp}(\mathsf{mk})$
If $i \neq i^\star$ Then
  $\mathsf{c} \leftarrow_\$ \mathsf{FE.Enc}((\mathsf{m}, \mathsf{k}), \mathsf{Mpk})$
If $i = i^\star$ Then
  Let $\{f_1, \ldots, f_n\} = \mathsf{TKList}$
  $\mathsf{k}' \leftarrow \mathsf{MAC.Col}(\mathsf{td}, f_1(\mathsf{m}), \ldots, f_n(\mathsf{m}), \mathsf{k})$
  $\mathsf{c} \leftarrow_\$ \mathsf{FE.Enc}((\mathsf{m}, \mathsf{k}'), \mathsf{Mpk})$
$i \leftarrow i + 1$; $\mathsf{List} \leftarrow \mathsf{List} \cup \{(i, \mathsf{m}, \mathsf{k})\}$
Return c

proc. Finalize$((y, \mathsf{t}), i, f)$:
If $i \neq i^\star$ Return F
Other steps as in VRF-CCA1

Fig. 19: Game_1 for VFE verifiability proof.

the collision algorithm of MAC and is included in the ciphertext. However, we still keep the original key k in List and hence also use this original key in **Finalize**. See Figure 19, where $Q := \mathbf{Q}^{\mathbf{Decrypt}}_{\mathsf{VFE},\mathcal{A}}(\lambda)$.

We show that the advantages of any PPT adversary $\mathcal{A}$ in Game_0 and Game_1 are negligibly different. To this end, we construct an algorithm $\mathcal{B}$ which interpolates between the two games and attacks the IND-CCA1 security of the underlying FE scheme. This algorithm is shown in Figure 20.

proc. Initialize(λ):
Get Mpk from $\mathbf{Initialize}_{\mathsf{FE}}(\lambda)$
$i^\star \leftarrow_\$ \{1, \ldots, Q\}$
$\mathsf{List} \leftarrow \{\}$; $\mathsf{TKList} \leftarrow \{\}$; $i \leftarrow 0$
$(\mathsf{td}, \mathsf{mk}) \leftarrow_\$ \mathsf{MAC.Setup}(1^\lambda)$
$\mathsf{hk} \leftarrow_\$ \mathsf{CRH.KeySp}(1^\lambda)$
Return $(\mathsf{Mpk}, \mathsf{mk}, \mathsf{hk})$

proc. Token(f):
$\mathsf{h}_f \leftarrow \mathsf{CRH.H}(\langle f \rangle, \mathsf{hk})$
$f^\star(x_1, x_2) := (f(x_1), \mathsf{MAC.Tag}((f(x_1), \mathsf{h}_f), x_2, \mathsf{Mpk}))$
Query $\mathbf{Token}_{\mathsf{FE}}(f^\star)$ to get TK
$\mathsf{TKList} \leftarrow \mathsf{TKList} \cup \{(f, \mathsf{h}_f)\}$
Return $(\mathsf{TK}, \mathsf{h}_f)$

proc. Decrypt(c, f):
$\mathsf{h}_f \leftarrow \mathsf{CRH.H}(\langle f \rangle, \mathsf{hk})$
$f^\star(x_1, x_2) := (f(x_1), \mathsf{MAC.Tag}((f(x_1), \mathsf{h}_f), x_2, \mathsf{Mpk}))$
Query $\mathbf{Decrypt}_{\mathsf{FE}}(\mathsf{c}, f^\star)$ to get m
Parse $(y, \mathsf{t}) \leftarrow \mathsf{m}$
Return (y, t)

proc. Encrypt(m):
$i \leftarrow i + 1$
$\mathsf{k} \leftarrow_\$ \mathsf{MAC.KeySp}(\mathsf{mk})$
If $i \neq i^\star$ Then
  $\mathsf{c} \leftarrow_\$ \mathsf{FE.Enc}((\mathsf{m}, \mathsf{k}), \mathsf{Mpk})$
If $i = i^\star$ Then
  Let $\{f_1, \ldots, f_n\} = \mathsf{TKList}$
  $\mathsf{k}' \leftarrow \mathsf{MAC.Col}(\mathsf{td}, f_1(\mathsf{m}), \ldots, f_n(\mathsf{m}), \mathsf{k})$
  Query $\mathbf{Left\text{-}Right}_{\mathsf{FE}}((\mathsf{m}, \mathsf{k}), (\mathsf{m}, \mathsf{k}'))$ to get c
$\mathsf{List} \leftarrow \mathsf{List} \cup \{(i, \mathsf{m}, \mathsf{k})\}$
Return c

proc. Finalize$((y, \mathsf{t}), i, f)$:
If $i \neq i^\star$ Call $\mathbf{Finalize}_{\mathsf{FE}}(1)$
If $(i, \star, \star) \notin \mathsf{List}$ Call $\mathbf{Finalize}_{\mathsf{FE}}(1)$
Let (m, k) be s.t. $(i, \mathsf{m}, \mathsf{k}) \in \mathsf{List}$
If $(f, \mathsf{h}_f) \notin \mathsf{TKList}$ Call $\mathbf{Finalize}_{\mathsf{FE}}(1)$
If $\mathsf{MAC.Ver}((y, \mathsf{h}_f), \mathsf{t}, \mathsf{k}, \mathsf{mk}) = \mathsf{F}$ Call $\mathbf{Finalize}_{\mathsf{FE}}(1)$
If $(f(\mathsf{m}) = y)$ Call $\mathbf{Finalize}_{\mathsf{FE}}(1)$
Call $\mathbf{Finalize}_{\mathsf{FE}}(0)$

Fig. 20: Algorithm $\mathcal{B}$ against IND-CCAx security of FE.

Note that adversary $\mathcal{B}$ will never query **Token** with an $f^\star$ such that $f^\star(\mathsf{m}, \mathsf{k}) \neq f^\star(\mathsf{m}, \mathsf{k}')$. To check this note the first component of the output will be equal to $f(\mathsf{m})$ and that k' is constructed in a chameleon way with respect to k, resulting in the same tag.

Now, if the bit chosen in the IND-CCA1 game is 0, algorithms $\mathcal{B}$ runs $\mathcal{A}$ in an environment identical to Game_0, and furthermore it will return 0 iff the adversary wins this game. We can therefore write:

$$\Pr\left[\mathsf{IND\text{-}CCA1}^{\mathcal{B}}_{\mathsf{FE}}(\lambda) \Rightarrow \mathsf{T} | b = 0\right] = \Pr\left[\mathsf{Game}_0{}^{\mathcal{A}}_{\mathsf{VFE}}(\lambda) \Rightarrow \mathsf{T}\right].$$

Furthermore, if the bit chosen in the IND-CCA1 game is 1, algorithm $\mathcal{B}$ runs $\mathcal{A}$ in an environment identical to Game_1 as k' is used in both cases. Furthermore, it return 1 iff the adversary loses this game. We therefore have:

$$\Pr\left[\mathsf{IND\text{-}CCA1}^{\mathcal{B}}_{\mathsf{FE}}(\lambda) \Rightarrow \mathsf{T} | b = 1\right] = 1 - \Pr\left[\mathsf{Game}_1{}^{\mathcal{A}}_{\mathsf{VFE}}(\lambda) \Rightarrow \mathsf{T}\right].$$

From definition of advantage, it follows that

$$\mathbf{Adv}^{\mathsf{ind\text{-}cca1}}_{\mathsf{FE},\mathcal{B}}(\lambda) = \Pr\left[\mathsf{Game}_0{}^{\mathcal{A}}_{\mathsf{VFE}}(\lambda) \Rightarrow \mathsf{T}\right] - \Pr\left[\mathsf{Game}_1{}^{\mathcal{A}}_{\mathsf{VFE}}(\lambda) \Rightarrow \mathsf{T}\right]$$

Based on adversary $\mathcal{A}$ in Game_1, we construct an adversary $\mathcal{C}$ against the UF-CMA security of MAC as shown in Figure 21.

proc. Initialize(λ):
Get mk from $\mathbf{Initialize}_{\mathsf{MAC}}(\lambda)$
$\mathsf{List} \leftarrow \{\}$; $\mathsf{TKList} \leftarrow \{\}$; $i \leftarrow 0$
$i^\star \leftarrow_\$ \{1, \ldots, Q\}$
$(\mathsf{Msk}, \mathsf{Mpk}) \leftarrow_\$ \mathsf{FE.Setup}(1^\lambda)$
$\mathsf{hk} \leftarrow_\$ \mathsf{CRH.KeySp}(1^\lambda)$
Return $(\mathsf{Mpk}, \mathsf{mk}, \mathsf{hk})$

proc. Encrypt(m):
$i \leftarrow i + 1$
$\mathsf{k} \leftarrow_\$ \mathsf{MAC.KeySp}(\mathsf{mk})$
If $i \neq i^\star$ Then
  $\mathsf{c} \leftarrow_\$ \mathsf{FE.Enc}((\mathsf{m}, \mathsf{k}), \mathsf{Mpk})$
  $\mathsf{List} \leftarrow \mathsf{List} \cup \{(i, \mathsf{m}, \mathsf{k})\}$
If $i = i^\star$ Then
  Let $\{f_1, \ldots, f_n\} = \mathsf{TKList}$
  Query $\mathbf{Collision}_{\mathsf{MAC}}(f_1(\mathsf{m}), \ldots, f_n(\mathsf{m}))$ to get k'
  $\mathsf{c} \leftarrow_\$ \mathsf{FE.Enc}((\mathsf{m}, \mathsf{k}'), \mathsf{Mpk})$
  $\mathsf{List} \leftarrow \mathsf{List} \cup \{(i, \mathsf{m}, \mathsf{k}')\}$
Return c

proc. Finalize$((y, \mathsf{t}), i, f)$:
If $i \neq i^\star$ Call $\mathbf{Finalize}_{\mathsf{MAC}}(\epsilon)$
If $(i, \star, \star) \notin \mathsf{List}$ Call $\mathbf{Finalize}_{\mathsf{MAC}}(\epsilon)$
Let m be s.t. $(i, \mathsf{m}, \mathsf{k}) \in \mathsf{List}$
If $(f, \mathsf{h}_f) \notin \mathsf{TKList}$ Call $\mathbf{Finalize}_{\mathsf{MAC}}(\epsilon)$
If $(f(\mathsf{m}) = y)$ Call $\mathbf{Finalize}_{\mathsf{MAC}}(\epsilon)$
Call $\mathbf{Finalize}_{\mathsf{MAC}}((y, \mathsf{h}_f), \mathsf{t})$

proc. Token(f):
$\mathsf{h}_f \leftarrow \mathsf{CRH.H}(\langle f \rangle, \mathsf{hk})$
$f^\star(x_1, x_2) := (f(x_1), \mathsf{MAC.Tag}((f(x_1), \mathsf{h}_f), x_2, \mathsf{Mpk}))$
$\mathsf{TK} \leftarrow_\$ \mathsf{VFE.TKGen}(f^\star, \mathsf{Msk})$
$\mathsf{TKList} \leftarrow \mathsf{TKList} \cup \{(f, \mathsf{h}_f)\}$
Return $(\mathsf{TK}, \mathsf{h}_f)$

proc. Decrypt(c, f):
$\mathsf{h}_f \leftarrow \mathsf{CRH.H}(\langle f \rangle, \mathsf{hk})$
$f^\star(x_1, x_2) := (f(x_1), \mathsf{MAC.Tag}((f(x_1), \mathsf{h}_f), x_2, \mathsf{Mpk}))$
$\mathsf{TK} \leftarrow_\$ \mathsf{VFE.TKGen}(f^\star, \mathsf{Msk})$
$\mathsf{m} \leftarrow \mathsf{VFE.Dec}(\mathsf{c}, \mathsf{TK})$
Parse $(y, \mathsf{t}) \leftarrow \mathsf{m}$ Return (y, t)

Fig. 21: Algorithm $\mathcal{C}$ against UF-CMA security of n-key chameleon MAC.

Adversary $\mathcal{C}$ runs $\mathcal{A}$ in the environment of Game_1. Furthermore, if $\mathcal{A}$ is successful then so is $\mathcal{C}$. This follows directly from the construction of our VFE scheme. Therefore:

$$\Pr\left[\mathsf{UF\text{-}CMA}^{\mathcal{C}}_{\mathsf{MAC}}(\lambda) \Rightarrow \mathsf{T}\right] = \Pr\left[\mathsf{Game}_1{}^{\mathcal{A}}_{\mathsf{VFE}}(\lambda) \Rightarrow \mathsf{T}\right].$$

The theorem now follows. □

F Input/Output Privacy of the DHE Construction

Theorem 8 (Input/output privacy). *Let $\mathcal{A}$ be a PPT adversary against the* TA-IND-CPA *security of the DHE scheme in Figure 6. Then there exists a PPT adversary $\mathcal{B}$ against the* IND-CPA *security of the HE scheme such that for all $\lambda \in \mathbb{N}$ we have:*

$$\mathbf{Adv}^{\mathsf{ta\text{-}ind\text{-}cpa}}_{\mathsf{DHE},\mathcal{A}}(\lambda) = \mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{FHE},\mathcal{B}}(\lambda).$$

proc. Initialize(λ):	**proc. Left-Right**$(\mathsf{m}_0, \mathsf{m}_1)$:	**proc. Finalize**(b'):
Get pk from $\mathbf{Initialize}_{\mathsf{FHE}}(\lambda)$	$(\mathsf{c}_0, \mathsf{aux}_0) \leftarrow_\$ \mathsf{VFE.Enc}(\mathsf{m}_0, \mathsf{Mpk})$	Call $\mathbf{Finalize}_{\mathsf{FHE}}(b')$
$(\mathsf{Msk}, \mathsf{Mpk}) \leftarrow_\$ \mathsf{VFE.Setup}(1^\lambda)$	$(\mathsf{c}_1, \mathsf{aux}_1) \leftarrow_\$ \mathsf{VFE.Enc}(\mathsf{m}_1, \mathsf{Mpk})$	
Return $(\mathsf{Msk}, \mathsf{Mpk}, \mathsf{pk})$	Query $\mathbf{Left\text{-}Right}_{\mathsf{FHE}}(\mathsf{c}_0, \mathsf{c}_1)$ to get $\mathsf{c}_{\mathsf{hom}}$	
	Return $\mathsf{c}_{\mathsf{hom}}$	

Fig. 22: Algorithm $\mathcal{B}$ against the IND-CPA security of FHE.

Proof. Let $\mathcal{A}$ be an algorithm as in the statement of the theorem. We construct algorithm $\mathcal{B}$ that attacks the semantic security of the underlying FHE scheme as shown in Figure 22.

From the description of $\mathcal{B}$ it is clear that:

$$\Pr\left[\mathsf{TA\text{-}IND\text{-}CPA}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T} | b = 1\right] = \Pr\left[\mathsf{IND\text{-}CPA}^{\mathcal{B}}_{\mathsf{FHE}}(\lambda) \Rightarrow \mathsf{T} | b = 1\right],$$

and

$$\Pr\left[\mathsf{TA\text{-}IND\text{-}CPA}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T} | b = 0\right] = \Pr\left[\mathsf{IND\text{-}CPA}^{\mathcal{B}}_{\mathsf{FHE}}(\lambda) \Rightarrow \mathsf{T} | b = 0\right].$$

The theorem follows. □

G Verifiability of the DHE Construction

Theorem 9 (Verifiability). *Let $\mathcal{A}$ be a PPT adversary against the* VRF-CCAx *security of the DHE scheme in Figure 6, placing at most $\mathbf{Q}^{\mathbf{Decrypt}}_{\mathsf{DHE},\mathcal{A}}(\lambda)$ queries to its* $\mathbf{Decrypt}$ *procedure. Then there exists a PPT adversary $\mathcal{B}$ against the* VRF-CCAx *security of the VFE scheme such that for all $\lambda \in \mathbb{N}$ we have:*

$$\mathbf{Adv}^{\mathsf{vrf\text{-}ccax}}_{\mathsf{DHE},\mathcal{A}}(\lambda) \leq (\mathbf{Q}^{\mathbf{Decrypt}}_{\mathsf{DHE},\mathcal{A}}(\lambda) + 1) \cdot \mathbf{Adv}^{\mathsf{vrf\text{-}ccax}}_{\mathsf{VFE},\mathcal{B}}(\lambda).$$

Proof. We prove this theorem via a sequence of games. Let $Q := \mathbf{Q}^{\mathbf{Decrypt}}_{\mathsf{DHE},\mathcal{A}}(\lambda)$. We define $\mathsf{Game}_{i^\star}$, for $i^\star = 0, \ldots, Q$, as follows. Game_0 is identical to the VRF-CCAx game. In $\mathsf{Game}_{i^\star}$, we change the behaviour of the decryption algorithm as follows:

- When $j > i^\star$, the j-th query to $\mathbf{Decrypt}$ is answered as in the VRF-CCAx game.
- For $j \leq i^\star$, and on input $(\mathsf{c}, i, \mathsf{h}_f)$:
  - We first strip the homomorphic layer in the queried ciphertext to obtain a value (y', t');
  - We then recover the ciphertext c' that we have provided to the adversary in encryption query i;
  - We also recover the token TK corresponding to the provided h_f.
  - Then we recompute the correct value of (y, t) by decrypting c' using TK;
  - If $(y, \mathsf{t}) \neq (y', \mathsf{t}')$ we simply reject the query.
  - Otherwise we return y.

We note that this alternative decryption procedure will only be inconsistent if (y', t') differs from the honestly computed (u, t), but still accepted by the verification algorithm of the VFE. Given the characteristics of the verification algorithm (uniqueness of tags), this can only occur if $y' \neq y$, which immediately leads to an attack on verifiability. More formally, we claim that any adversary $\mathcal{A}$ whose behaviour changes visibly between $\mathsf{Game}_{i^\star}$ and $\mathsf{Game}_{i^\star+1}$ can be converted into an attacker of VFE verifiability $\mathcal{B}$ as shown in Figure 23.

We observe that $\mathcal{B}$'s simulation of both games is perfect until $\mathcal{A}$ performs the $(i^\star + 1)$-st query: this query might be rejected in $\mathsf{Game}_{i^\star+1}$ whereas in $\mathsf{Game}_{i^\star}$ adversary $\mathcal{A}$ could receive a decryption value. Furthermore, if this event occurs $\mathcal{B}$ returns a valid forgery in the verifiability game. Hence, by the fundamental lemma of game hopping we have:

$$\Pr[\mathsf{Game}_{i^\star}{}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}] - \Pr[\mathsf{Game}_{i^\star+1}{}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}] \leq \Pr[\mathsf{VRF\text{-}CCAx}^{\mathcal{B}}_{\mathsf{VFE}}(\lambda) \Rightarrow \mathsf{T}].$$

proc. $\mathbf{Initialize}(\lambda)$:
Get Mpk from $\mathbf{Initialize}_{\mathsf{VFE}}(\lambda)$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow_{\$} \mathsf{FHE.Gen}(1^\lambda)$
$\mathsf{TKList}' \leftarrow \{\}$; $\mathsf{List}' \leftarrow \{\}$
$i, j \leftarrow 0$; $X \leftarrow \epsilon$
Return $(\mathsf{Mpk}, \mathsf{pk})$

proc. $\mathbf{Token}(f)$:
Query $\mathbf{Token}_{\mathsf{VFE}}(f)$ to get $(\mathsf{TK}, \mathsf{h}_f)$
$\mathsf{TKList}' \leftarrow \mathsf{TKList}' \cup \{(f, \mathsf{h}_f, \mathsf{TK})\}$
Return $(\mathsf{TK}, \mathsf{h}_f)$

proc. $\mathbf{Decrypt}(\mathsf{c}', i, \mathsf{h}_f)$:
$j \leftarrow j + 1$
If $(i, \star, \star) \notin \mathsf{List}'$ Return $\perp$
If $(f, \mathsf{h}_f, \mathsf{TK}) \notin \mathsf{TKList}'$ Return $\perp$
Let (m, c) be s.t. $(i, \mathsf{m}, \mathsf{c}) \in \mathsf{List}'$
$\mathsf{m} \leftarrow \mathsf{FHE.Dec}(\mathsf{c}', \mathsf{sk})$
Parse $(y', \mathsf{t}') \leftarrow \mathsf{m}$
$(y, \mathsf{t}) \leftarrow \mathsf{VFE.Dec}(\mathsf{c}, \mathsf{TK})$
If $(y, \mathsf{t}) \neq (y', \mathsf{t}')$
  If $j \leq i^\star$ Return $\perp$
  If $j = i^\star + 1$ Then
    $X \leftarrow ((y', \mathsf{t}'), i, f)$
    Return $\perp$
  If $j > i^\star + 1$ Return $\perp$
Return y

proc. $\mathbf{Encrypt}(\mathsf{m})$:
Query $\mathbf{Encrypt}_{\mathsf{VFE}}(\mathsf{m})$ to get (c)
$\mathsf{c}_{\mathsf{hom}} \leftarrow_{\$} \mathsf{FHE.Enc}(\mathsf{c}, \mathsf{pk})$
$i \leftarrow i + 1$
$\mathsf{List}' \leftarrow \mathsf{List}' \cup \{(i, \mathsf{m}, \mathsf{c})\}$
Return $\mathsf{c}_{\mathsf{hom}}$

proc. $\mathbf{Finalize}(\mathsf{c}^\star, i, \mathsf{h}_f)$:
Call $\mathbf{Finalize}_{\mathsf{VFE}}(X)$

Fig. 23: Algorithm $\mathcal{B}$ against the VRF-CCAx security of VFE, interpolating between games $i^\star$ and $i^\star + 1$.

It follows that:

$$\Pr[\mathsf{VRF\text{-}CCAx}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}] - \Pr[\mathsf{Game}_Q{}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}] \leq \mathbf{Q}^{\mathbf{Decrypt}}_{\mathsf{DHE},\mathcal{A}}(\lambda) \cdot \Pr[\mathsf{VRF\text{-}CCAx}^{\mathcal{B}}_{\mathsf{VFE}}(\lambda) \Rightarrow \mathsf{T}].$$

To complete the proof, we note that any attacker against Game_Q can immediately be converted into an attacker on the verifiability of VFE. We define this attacker $\mathcal{C}$ as shown in Figure 24.

The theorem follows from the observation that $\mathcal{C}$'s simulation is perfect and whenever $\mathcal{A}$ wins Game_Q, C succeeds in attacking verifiability:

$$\Pr[\mathsf{Game}_Q{}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}] = \Pr[\mathsf{VRF\text{-}CCAx}^{\mathcal{C}}_{\mathsf{VFE}}(\lambda) \Rightarrow \mathsf{T}].$$

proc. $\mathbf{Initialize}(\lambda)$:
Get Mpk from $\mathbf{Initialize}_{\mathsf{VFE}}(\lambda)$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow_{\$} \mathsf{FHE.Gen}(1^\lambda)$
$\mathsf{TKList}' \leftarrow \{\}$; $\mathsf{List}' \leftarrow \{\}$
$i \leftarrow 0$; $X \leftarrow \epsilon$
Return $(\mathsf{Mpk}, \mathsf{pk})$

proc. $\mathbf{Token}(f)$:
Query $\mathbf{Token}_{\mathsf{VFE}}(f)$ to get $(\mathsf{TK}, \mathsf{h}_f)$
$\mathsf{TKList}' \leftarrow \mathsf{TKList}' \cup \{(f, \mathsf{h}_f, \mathsf{TK})\}$
Return $(\mathsf{TK}, \mathsf{h}_f)$

proc. $\mathbf{Decrypt}(\mathsf{c}', i, \mathsf{h}_f)$:
If $(i, \star, \star) \notin \mathsf{List}'$ Return $\perp$
If $(f, \mathsf{h}_f, \mathsf{TK}) \notin \mathsf{TKList}'$ Return $\perp$
Let (m, c) be s.t. $(i, \mathsf{m}, \mathsf{c}) \in \mathsf{List}'$
$\mathsf{m} \leftarrow \mathsf{FHE.Dec}(\mathsf{c}', \mathsf{sk})$
Parse $(y', \mathsf{t}') \leftarrow \mathsf{m}$
$(y, \mathsf{t}) \leftarrow \mathsf{VFE.Dec}(\mathsf{c}, \mathsf{TK})$
If $(y, \mathsf{t}) \neq (y', \mathsf{t}')$ Return $\perp$
Return y

proc. $\mathbf{Encrypt}(\mathsf{m})$:
Query $\mathbf{Encrypt}_{\mathsf{VFE}}(\mathsf{m})$ to get (c)
$\mathsf{c}_{\mathsf{hom}} \leftarrow_{\$} \mathsf{FHE.Enc}(\mathsf{c}, \mathsf{pk})$
$i \leftarrow i + 1$
$\mathsf{List}' \leftarrow \mathsf{List}' \cup \{(i, \mathsf{m}, \mathsf{c})\}$
Return $\mathsf{c}_{\mathsf{hom}}$

proc. $\mathbf{Finalize}(\mathsf{c}', i, \mathsf{h}_f)$:
If $(i, \star, \star) \notin \mathsf{List}'$ Then
  Call $\mathbf{Finalize}_{\mathsf{VFE}}(\epsilon)$
If $(f, \mathsf{h}_f, \mathsf{TK}) \notin \mathsf{TKList}'$ Then
  Call $\mathbf{Finalize}_{\mathsf{VFE}}(\epsilon)$
$\mathsf{m} \leftarrow \mathsf{FHE.Dec}(\mathsf{c}', \mathsf{sk})$
Parse $(y', \mathsf{t}') \leftarrow \mathsf{m}$
Call $\mathbf{Finalize}_{\mathsf{VFE}}((y', \mathsf{t}'), i, f)$

Fig. 24: Algorithm $\mathcal{C}$ against the VRF-CCAx security of VFE.

□

H Evaluation Security of the DHE Construction

Theorem 10 (Evaluation security). *Let $\mathcal{A}$ be a PPT adversary against the* IND-EVALx *security of the DHE scheme in Figure 6. Then there exists a PPT adversary $\mathcal{B}$ against* IND-CCAx *security of the FE scheme that, for all*

$\lambda \in \mathbb{N}$ *we have:*

$$\mathbf{Adv}^{\mathsf{ind\text{-}evalx}}_{\mathsf{DHE},\mathcal{A}}(\lambda) = \mathbf{Adv}^{\mathsf{ind\text{-}ccax}}_{\mathsf{VFE},\mathcal{B}}(\lambda).$$

Proof. Let $\mathcal{A}$ be an algorithm as in the statement of the theorem. We construct an algorithm $\mathcal{B}$ against the IND-CCAx security of VFE as shown in Figure 25.

proc. Initialize(λ):	**proc. Evaluate**(c, f):	**proc. Left-Right**$(\mathsf{m}_0, \mathsf{m}_1)$:
Get Mpk from $\mathbf{Initialize}_{\mathsf{VFE}}(\lambda)$	$\mathsf{c}' \leftarrow \mathsf{FHE.Dec}(\mathsf{c}, \mathsf{sk})$	Query $\mathbf{Left\text{-}Right}_{\mathsf{VFE}}(\mathsf{m}_0, \mathsf{m}_1)$ to get $(\mathsf{c}', \mathsf{aux})$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow_{\$} \mathsf{FHE.Gen}(1^\lambda)$	Query $\mathbf{Decrypt}_{\mathsf{VFE}}(\mathsf{c}', f)$ to get m	$\mathsf{c} \leftarrow_{\$} \mathsf{FHE.Enc}(\mathsf{c}', \mathsf{pk})$
Return $(\mathsf{Mpk}, \mathsf{sk}, \mathsf{pk})$	$\mathsf{c}'' \leftarrow_{\$} \mathsf{FHE.Enc}(\mathsf{m}, \mathsf{pk})$	Return $(\mathsf{c}, \mathsf{aux})$
	Return c''	
proc. Token(f):		**proc. Finalize**(b'):
Query $\mathbf{Token}_{\mathsf{VFE}}(f)$ to get $(\mathsf{TK}, \mathsf{h}_f)$		Call $\mathbf{Finalize}_{\mathsf{VFE}}(b')$
Return $(\mathsf{TK}, \mathsf{h}_f)$		

Fig. 25: Algorithm $\mathcal{B}$ against the IND-CCAx security of VFE.

From the description of $\mathcal{B}$ it is clear that if $\mathcal{A}$ is a legitimate adversary then so is $\mathcal{B}$, and furthermore if $\mathcal{A}$ does not extract a token for an illegal function f such that $f(\mathsf{m}_0) \neq f(\mathsf{m}_1)$, then $\mathcal{B}$ also does not. This means that

$$\Pr\left[\mathsf{IND\text{-}EVALx}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T} | b = 1\right] = \Pr\left[\mathsf{IND\text{-}CCAx}^{\mathcal{B}}_{\mathsf{VFE}}(\lambda) \Rightarrow \mathsf{T} | b = 1\right],$$

and

$$\Pr\left[\mathsf{IND\text{-}EVALx}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T} | b = 0\right] = \Pr\left[\mathsf{IND\text{-}CCAx}^{\mathcal{B}}_{\mathsf{VFE}}(\lambda) \Rightarrow \mathsf{T} | b = 0\right].$$

This proves the theorem. □

I Input/Output Privacy of the VC Construction

Theorem 11 (Input/output privacy). *Let $\mathcal{A}$ be a PPT adversary against the* IND-CPA *security of the VC scheme in Figure 7. Then there exists a PPT adversary $\mathcal{B}$ against the* TA-IND-CPA *security of the underlying DHE scheme such that for any $f \in \mathsf{FunSp}(1^\lambda)$ and for all $\lambda \in \mathbb{N}$ we have:*

$$\mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{f,\mathsf{VC},\mathcal{A}}(\lambda) = \mathbf{Adv}^{\mathsf{ta\text{-}ind\text{-}cpa}}_{\mathsf{DHE},\mathcal{B}}(\lambda).$$

Proof. Let $\mathcal{A}$ be an algorithm as in the statement of the theorem. We construct the algorithm $\mathcal{B}$ against the TA-IND-CPA security of DHE as shown in Figure 26.

proc. Initialize(f, λ):	**proc. Left-Right**$(\mathsf{m}_0, \mathsf{m}_1)$:
Get $(\mathsf{Msk}, \mathsf{Mpk}, \mathsf{pk})$ from $\mathbf{Initialize}_{\mathsf{DHE}}(\lambda)$	Query $\mathbf{Left\text{-}Right}_{\mathsf{DHE}}(\mathsf{m}_0, \mathsf{m}_1)$ to get c
$(\mathsf{TK}, \mathsf{h}_f) \leftarrow_{\$} \mathsf{DHE.TKGen}(f, \mathsf{Msk})$	Return c
Return $(\mathsf{TK}, \mathsf{pk})$	
	proc. Finalize(b'):
proc. PubProbGen(m):	Call $\mathbf{Finalize}_{\mathsf{DHE}}(b')$
$(\mathsf{c}, \mathsf{aux}) \leftarrow_{\$} \mathsf{DHE.Enc}(\mathsf{m}, \mathsf{pk})$	
Return c	

Fig. 26: Algorithm $\mathcal{B}$ against the TA-IND-CPA security of DHE.

From the description of $\mathcal{B}$ it is clear that:

$$\Pr\left[\mathsf{IND\text{-}CPA}^{\mathcal{A}}_{f,\mathsf{VC}}(\lambda) \Rightarrow \mathsf{T} | b = 1\right] = \Pr\left[\mathsf{TA\text{-}IND\text{-}CPA}^{\mathcal{B}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T} | b = 1\right],$$

and

$$\Pr\left[\mathsf{IND\text{-}CPA}^{\mathcal{A}}_{f,\mathsf{VC}}(\lambda) \Rightarrow \mathsf{T} | b = 0\right] = \Pr\left[\mathsf{TA\text{-}IND\text{-}CPA}^{\mathcal{B}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T} | b = 0\right].$$

The theorem follows. □

J Verifiability of the VC Construction

Theorem 12 (Verifiability). *Let $\mathcal{A}$ be a PPT adversary against the* VRF-CCA1 *security of the VC scheme in Figure 7. Then there exists a PPT adversary $\mathcal{B}$ against the* VRF-CCA1 *security of the underlying DHE scheme which makes only a single (non-adaptive)* **Token** *query, and such that for any $f \in \mathsf{FunSp}(1^\lambda)$ and for all $\lambda \in \mathbb{N}$ we have:*

$$\mathbf{Adv}^{\mathsf{vrf\text{-}cca1}}_{f,\mathsf{VC},\mathcal{A}}(\lambda) = \mathbf{Adv}^{\mathsf{vrf\text{-}cca1}}_{\mathsf{DHE},\mathcal{B}}(\lambda).$$

Proof. Let $\mathcal{A}$ be an algorithm as in the statement of the theorem. We construct the algorithm $\mathcal{B}$ attacking VRF-CCA1 security of DHE as shown in Figure 27.

proc. Initialize(f, λ):	**proc. PubVerify**(c, i):
Get $(\mathsf{Mpk}, \mathsf{pk})$ from $\mathbf{Initialize}_{\mathsf{DHE}}(\lambda)$	Query $\mathbf{Decrypt}_{\mathsf{DHE}}(\mathsf{c}, i, \mathsf{h}_f)$ to get m
Query $\mathbf{Token}_{\mathsf{DHE}}(f)$ to get $(\mathsf{TK}, \mathsf{h}_f)$	Return m
Return $(\mathsf{TK}, \mathsf{pk})$	
	proc. Finalize$(\mathsf{c}^\star, i)$:
proc. PubProbGen(m):	Call $\mathbf{Finalize}_{\mathsf{DHE}}(\mathsf{c}^\star, i, \mathsf{h}_f)$
Query $\mathbf{Encrypt}_{\mathsf{DHE}}(\mathsf{m})$ to get c	
Return c	

Fig. 27: Algorithm $\mathcal{B}$ attacking VRF-CCA1 security of DHE.

From the description of $\mathcal{B}$ it is clear that:

$$\Pr\left[\mathsf{VRF\text{-}CCA1}^{\mathcal{A}}_{f,\mathsf{VC}}(\lambda) \Rightarrow \mathsf{T}\right] = \Pr\left[\mathsf{VRF\text{-}CCA1}^{\mathcal{B}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}\right],$$

as $\mathcal{B}$ perfectly simulates the VRF-CCA1 for $\mathcal{A}$. Note that $\mathcal{B}$ makes exactly one query to **Token** before its **Encrypt** queries. The theorem follows. □