Targeted Malleability: Homomorphic Encryption for Restricted Computations

Dan Boneh* Gil Segev† Brent Waters‡

Abstract

We put forward the notion of *targeted malleability*: given a homomorphic encryption scheme, in various scenarios we would like to restrict the homomorphic computations one can perform on encrypted data. We introduce a precise framework, generalizing the foundational notion of *non-malleability* introduced by Dolev, Dwork, and Naor (SICOMP '00), ensuring that the malleability of a scheme is targeted only at a specific set of "allowable" functions.

In this setting we are mainly interested in the efficiency of such schemes as a function of the number of repeated homomorphic operations. Whereas constructing a scheme whose ciphertext grows linearly with the number of such operations is straightforward, obtaining more realistic (or merely non-trivial) length guarantees is significantly more challenging.

We present two constructions that transform any homomorphic encryption scheme into one that offers targeted malleability. Our constructions rely on standard cryptographic tools and on succinct non-interactive arguments, which are currently known to exist in the standard model based on variants of the knowledge-of-exponent assumption. The two constructions offer somewhat different efficiency guarantees, each of which may be preferable depending on the underlying building blocks.

Keywords: Homomorphic encryption, Non-malleable encryption.

*Stanford University. Supported by NSF, DARPA, and AFOSR.

†Microsoft Research, Mountain View, CA 94043, USA.

‡University of Texas at Austin. Supported by NSF CNS-0716199, CNS-0915361, and CNS-0952692, DARPA PROCEED, Air Force Office of Scientific Research (AFO SR) MURI, DHS Grant 2006-CS-001-000001-02, and the Sloan Foundation.

1 Introduction

Fully homomorphic encryption [RAD78, Gen09, SV10, vDGH+10] is a remarkable development in cryptography enabling anyone to compute arbitrary functions on encrypted data. In many settings, however, the data owner may wish to restrict the class of homomorphic computations to a certain set $\mathcal{F}$ of allowable functions. In this paper we put forward the notion of "targeted malleability": given an encryption scheme that supports homomorphic operations with respect to some set of functions $\mathcal{F}$, we would like to ensure that the malleability of the scheme is targeted only at the set $\mathcal{F}$. That is, it should not be possible to apply any homomorphic operation other than the ones in $\mathcal{F}$.

Enforcing targeted malleability can be simply done by requiring the entity performing the homomorphic operation to embed a proof in the ciphertext showing that the ciphertext was computed using an allowable function. The decryptor then verifies the proof before decrypting the ciphertext, and outputs $\perp$ if the proof is invalid. Unfortunately, as the homomorphic operation is repeated the number of proofs grows making the ciphertext grow at least *linearly* with the number of repeated homomorphic operations. It is not difficult to see that targeted malleability with a linear-size ciphertext is trivial to construct: Use any non-malleable encryption scheme, and embed in the ciphertext a description of all the functions being computed. The decryptor decrypts the original ciphertext and applies the embedded functions to it (verifying, of course, that these functions are in the allowable set).

Minimizing ciphertext expansion. Targeted malleability is much harder to construct once we require that ciphertext growth is at most sub-linear in the number of repeated homomorphic operations. Our goal is to construct systems where even after t applications of the homomorphic operation the ciphertext length does not increase much. In our main construction we are able to completely shift the dependence on t from the ciphertext to the public key: the ciphertext size is essentially independent of t. This is a natural goal since public keys are typically much more static than ciphertexts which are frequently generated and transmitted.

Motivation. While targeted malleability is an interesting concept in its own right, it has many applications in cryptography and beyond. We give a few illustrative examples:

- A spam filter implemented in a mail server adds a `spam` tag to encrypted emails whose content satisfies a certain spam predicate. The filter should be allowed to run the spam predicate, but should not modify the email contents. In this case, the set of allowable functions $\mathcal{F}$ would be the set of allowable spam predicates and nothing else. As email passes from one server to the next each server homomorphically computes its spam predicate on the encrypted output of the previous server. Each spam filter in the chain can run its chosen spam predicate and nothing else.

- More generally, in a distributed system users initiate encrypted requests to various servers. To service a request a server may need to contact another server and that server may need to contact another, resulting in a chain of messages from server to server until the transaction is fulfiled. Each server along the way has an allowed set of operations it can apply to a recieved message and it should be unable to apply any operation outside this approved set.

- In a voting system based on homomorphic encryption (e.g. [CGS97]) voters take turns incrementing an encrypted vote tally using a homomorphic operation. They are only allowed to increase the encrypted tally by 1 (indicating a vote for the candidate) or by 0 (indicating a

no vote for the candidate). In elections where each voter votes for one of ℓ candidates, voters modify the encrypted tallies by adding an ℓ-bit vector, where exactly one entry is 1 and the rest are all 0's. They should be unable to modify the counters in any other way.

In all these examples there is a need to repeatedly apply a restricted homomorphic operation on encrypted data. Limiting ciphertext expansion is highly desirable.

1.1 Our Contributions

We begin by introducing a precise framework for modeling targeted malleability. Our notion of security generalizes the foundational one of non-malleability due to Dolev, Dwork, and Naor [DDN00], and is also inspired by the refinements of Bellare and Sahai [BS99], and Pass, Shelat, and Vaikuntanathan [PSV07]. Given a public-key encryption scheme that is homomorphic with respect to a set of functions $\mathcal{F}$ we would like to capture the following intuitive notion of security[1]: For any efficient adversary that is given an encryption c of a message m and outputs an encryption c' of a message m', it should hold that either (1) m' is independent of m, (2) $c' = c$ (and thus $m' = m$), or (3) c' is obtained by repeatedly applying the homomorphic evaluation algorithm on c using functions $f_1, \ldots, f_\ell \in \mathcal{F}$. The first two properties are the standard ones for non-malleable encryption, and the third property captures our new notion of targeted malleability: we would like to target the malleability of the scheme only at the class $\mathcal{F}$ (we note that by setting $\mathcal{F} = \emptyset$ we recover the standard definition of non-malleability)[2]. We consider this notion of security with respect to both chosen-plaintext attacks (CPA) and a-priori chosen-ciphertext attacks (CCA1)[3].

We emphasize that we do not make the assumption that the set of functions $\mathcal{F}$ is closed under composition. In particular, our approach is sufficiently general to allow targeting the malleability of a scheme at any *subset* $\mathcal{F}' \subseteq \mathcal{F}$ of the homomorphic operations that are supported by the scheme. This is significant, for example, when dealing with fully homomorphic schemes, where any set of functions is in fact a subset of the supported homomorphic operations (see Section 1.3 for more details).

Next, we present two general transformations that transform any homomorphic encryption scheme into one that enjoys targeted malleability for a limited number of repeated homomorphic operations. The resulting schemes are secure even in the setting of a-priori chosen-ciphertext attacks (CCA1). The two constructions offer rather different trade-offs in terms of efficiency. In this overview we focus on our first construction, as it already captures the main ideas underlying our methodology.

1.2 Overview of Our Approach

Our approach is based on bridging between two seemingly conflicting goals: on one hand, we would like to turn the underlying homomorphic scheme into a somewhat non-malleable one, whereas on the other hand we would like to preserve its homomorphic properties. We demonstrate that the Naor-Yung "double encryption" paradigm for non-malleability [NY90, DDN00, Sah99, Lin06] can be utilized to obtain an interesting balance between these two goals. The structure of ciphertexts in our construction follows the latter paradigm: a ciphertext is a 3-tuple (c_0, c_1, π) containing two

[1] For simplicity we focus here on univariate functions and refer the reader to Section 3 for the more general case of multivariate functions

[2] We assume in this informal discussion that the adversary outputs a *valid* ciphertext, but our notion of security in fact considers the more general case – see Section 3.

[3] See Section 6 for a discussion on a-posteriori chosen-ciphertext attacks (CCA2) in the setting of homomorphic encryption, following the work of Prabhakaran and Rosulek [PR08].

encryptions of the same message using the underlying encryption scheme under two different keys along with a proof π that the ciphertext is well formed. For ciphertexts that are produced by the encryption algorithm, π is a non-interactive zero-knowledge proof, and for ciphertexts that are produced by the homomorphic evaluation algorithm, π is a succinct non-interactive argument that need not be zero-knowledge.

Specifically, the public key of the scheme consists of two public keys, pk_0 and pk_1, of the underlying homomorphic scheme, a common reference string for a non-interactive zero-knowledge proof system, and t common reference strings for succinct non-interactive argument systems (where t is a predetermined upper bound on the number of repeated homomorphic operations that can be applied to a ciphertext produced by the encryption algorithm). The secret key consists of the corresponding secret keys sk_0 and sk_1. For encrypting a message we encrypt it under each of pk_0 and pk_1, and provide a non-interactive zero-knowledge proof that the resulting two ciphertexts are indeed encryptions of the same message. Thus, a ciphertext that is produced by the encryption algorithm has the form $(c_0, c_1, \pi_{\mathsf{ZK}})$.

The homomorphic evaluation algorithm preserves the "double encryption" invariant. Specifically, given a ciphertext $(c_0, c_1, \pi_{\mathsf{ZK}})$ that was produced by the encryption algorithm and a function $f \in \mathcal{F}$, the homomorphic evaluation algorithm first applies the homomorphic evaluation algorithm of the underlying encryption scheme to each of c_0 and c_1. That is, it computes $c_0^{(1)} = \mathsf{HomEval}_{pk_0}(c_0, f)$ and $c_1^{(1)} = \mathsf{HomEval}_{pk_1}(c_1, f)$. Then, it computes a succinct non-interactive argument $\pi^{(1)}$ to the fact that there exist a function $f \in \mathcal{F}$ and a ciphertext $(c_0, c_1, \pi_{\mathsf{ZK}})$, such that π_{ZK} is accepted by the verifier of the non-interactive zero-knowledge proof system, and that $c_0^{(1)}$ and $c_1^{(1)}$ are generated from c_0 and c_1 using f as specified. We denote the language of the corresponding argument system by $L^{(1)}$, and the resulting ciphertext is of the form $c^{(1)} = \left(1, c_0^{(1)}, c_1^{(1)}, \pi^{(1)}\right)$. We point out that the usage of *succinct* arguments enables us to prevent the length of ciphertexts from increasing significantly.

More generally, given a ciphertext of the form $c^{(i)} = \left(i, c_0^{(i)}, c_1^{(i)}, \pi^{(i)}\right)$, the homomorphic evaluation algorithm follows the same methodology for producing a ciphertext of the same form $c^{(i+1)} = \left(i+1, c_0^{(i+1)}, c_1^{(i+1)}, \pi^{(i+1)}\right)$ using a succinct non-interactive argument system for a language $L^{(i+1)}$ stating that there exist a function $f \in \mathcal{F}$ and a ciphertext $c^{(i)}$ that is well-formed with respect to $L^{(i)}$, which were used for generating the current ciphertext $c^{(i+1)}$.

On the proof of security. Given an adversary that breaks the targeted malleability of our construction, we construct an adversary that breaks the security of (at least one of) the underlying building blocks. As in [NY90, DDN00, Sah99, Lin06], we show that this boils down to having a simulator that is able to decrypt a ciphertext while having access to only one of the secret keys sk_0 and sk_1. This, in turn, enables the simulator to attack the public key pk_b for which sk_b is not known, where $b \in \{0,1\}$. For satisfying our notion of security, however, such a simulator will not only have to decrypt a ciphertext, but to also recover a "certification chain" demonstrating that the ciphertext was produced by repeatedly applying the homomorphic evaluation algorithm. That is, given a well-formed ciphertext $c^{(i)} = \left(i, c_0^{(i)}, c_1^{(i)}, \pi^{(i)}\right)$, the simulator needs to generate a "certification chain" for $c^{(i)}$ of the form $\left(c^{(0)}, f^{(0)}, \ldots, c^{(i-1)}, f^{(i-1)}, c^{(i)}\right)$, where:

1. $c^{(0)}$ is an output of the encryption algorithm, which can be decrypted while knowing only one of sk_0 and sk_1.

2. For every $j \in \{1, \ldots, i\}$ it holds that $c^{(j)}$ is obtained by applying the homomorphic evaluation

algorithm on $c^{(j-1)}$ and $f^{(j-1)}$.

For this purpose, we require that the argument systems used in the construction exhibit the following "knowledge extraction" property: for every efficient malicious prover P^* there exists an efficient "knowledge extractor" $\mathsf{Ext}_{\mathsf{P}^*}$, such that whenever P^* outputs a statement x and an argument π that are accepted by the verifier, $\mathsf{Ext}_{\mathsf{P}^*}$ when given the random coins of P^* can in fact produce a witness w to the validity of x with all but a negligible probability.

By repeatedly applying such extractors the simulator is able to produce a certification chain. Then, given that the initial ciphertext $c^{(0)}$ is well-formed (i.e., the same messages is encrypted under pk_0 and pk_1), it can be decrypted using only one of the corresponding secret keys.

An alternative trade-off. In our first construction, the length of the ciphertext is essentially independent of t, and the public key consists of $t+1$ common reference strings. In our second construction the number of common reference strings in the public key is only $\log t$, and a ciphertext now consists of $\log t$ ciphertexts of the underlying homomorphic scheme and $\log t$ succinct arguments. Such a trade-off may be preferable over the one offered by our first construction, for example, when using argument systems that are tailored to the $\mathcal{NP}$ languages under considerations, or when it is not possible to use the same common reference string for all argument systems (depending, of course, on the length of the longest common reference strings).

The main idea underlying this construction is that the arguments computed by the homomorphic evaluation algorithm form a tree structure instead of a path structure. Specifically, instead of using t argument systems, we use only $d = \log t$ argument systems where the i-th one is used for arguing the well-formedness of a ciphertext after 2^i repeated homomorphic operations.

Succinct extractable arguments. As explained above, our construction hinges on the existence of succinct non-interactive argument systems that exhibit a knowledge extractor capable of extracting a witness from any successful prover. Gentry and Wichs [GW11] recently showed that no sub-linear non-interactive argument system can be proven secure by a black-box reduction to a falsifiable assumption. Fortunately, while we need succinct arguments, their lengths need not be sub-linear. It suffices for our purposes that arguments are shorter by only a multiplicative constant factor (say, 1/4) than the length of the witness, and therefore the negative result of Gentry and Wichs does not apply to our settings. Nevertheless, all known argument systems that satisfy our needs are either set in the random oracle model or are based on non-falsifiable assumptions in the sense of Naor [Nao03].

The first such system was constructed by Micali [Mic00] using the PCP theorem. Computational soundness is proved in the random oracle model [BR93] and the length of the proofs is essentially independent of the length of the witness. Valiant [Val08] observed that the system is extractable as needed for our proof of security. Unfortunately, we inherently cannot use an argument system set in the random oracle model. To see why, consider a fresh ciphertext c which is an encryption of message m. After the first homomorphic operation we obtain a new ciphertext c' containing a proof π showing that c' is an encryption of $f(m)$ for some allowable function $f \in \mathcal{F}$. Verifying π requires access to the random oracle. Now, consider the second homomorphic operation resulting in c''. The proof embedded in c'' must now prove, among other things, that there exists a valid proof π showing that c' is a well-formed ciphertext. But since π's verifier queries the random oracle, this statement is in $\mathcal{NP}^{\mathcal{O}}$ where $\mathcal{O}$ is a random oracle. Since PCPs do not relativize, it seems that Micali's system cannot be used for our purpose. In fact, there are no known succinct argument systems for proving statements in $\mathcal{NP}^{\mathcal{O}}$. This issue was also pointed out by Chiesa

and Tromer [CT10] in a completely different context, who suggested to overcome this difficulty by providing each prover with a smartcard implementing a specific oracle functionality.

Instead, we use a recent succinct non-interactive argument system due to Groth [Gro10] (see also the refinement by Lipmaa [Lip11]). Soundness is based on a variant of the "knowledge of exponent assumption," a somewhat non-standard assumption (essentially stating that the required extractor exists by assumption, backed up with evidence in the generic group model) . This class of assumptions was introduced by Damgård [Dam91] and extended by Bellare and Palacio [BP04b]. Interestingly, Bellare and Palacio [BP04a] succeeded in falsifying one such assumption using the Decision Diffie-Hellman problem. We note that while Groth's argument system is even zero-knowledge, we primarily use the soundness property of the system (see the discussion in Section 6 on exploiting its zero-knowledge property).

1.3 Related Work

The problem of providing certain non-malleability properties for homomorphic encryption schemes was studied by Prabhakaran and Rosulek [PR08]. As a positive result, they presented a variant of the Cramer-Shoup encryption scheme [CS98] that provably supports linear operations and no other operations. There are two main differences between our work and the work of Prabhakaran and Rosulek: (1) their framework only considers sets of allowable functions that are *closed under composition*, and (2) their framework does not prevent ciphertext expansion during repeated applications of the homomorphic operation, whereas this is a key goal for our work.

In our work we do not make the assumption that the set of allowable functions $\mathcal{F}$ is closed under composition. As already discussed, one of the advantages of avoiding this assumption (other than the obvious advantage of capturing a wider class of homomorphic schemes) is that we are in fact able to target the malleability of a scheme at any subset $\mathcal{F}' \subseteq \mathcal{F}$ of its supported homomorphic operations (which may be determined by the specific application in which the scheme is used), and this is especially significant when dealing with fully homomorphic schemes. Another advantage is the ability to limit the number of repeated homomorphic operations.

We note that when assuming that the set of functions $\mathcal{F}$ is closed under composition, there is in fact a trivial solution: For encrypting a message m compute $(\mathsf{Enc}_{pk}(m), \mathsf{id})$ using any non-malleable encryption scheme, where id is the identity function. Then, the homomorphic evaluation algorithm on input a ciphertext (c, f_1) and a function $f_2 \in \mathcal{F}$ simply outputs $(c, f_2 \circ f_1)$ (where $\circ$ denotes composition of functions). In this light, Prabhakaran and Rosulek focused on formalizing a meaningful notion of security for a-posteriori chosen-ciphertext attacks (CCA2), following previous relaxations of such attacks [ADR02, CKN03, Gro04, PR07]. This is orthogonal to our setting in which the issue of avoiding a blow-up in the length of the ciphertext makes the problem challenging already for chosen-plaintext attacks.

Finally, we note that targeted malleability shares a somewhat similar theme with the problem of outsourcing a computation in a verifiable manner from a computationally-weak client to a powerful server (see, for example, [GKR08, GGP10, CKV10, AIK10] and the references therein). In both settings the main goal from the security aspect is to guarantee that a "correct" or an "allowable" computation is performed. From the efficiency aspect, however, the two settings significantly differ: whereas for targeted malleability our main focus is to prevent a blow-up in the length of the ciphertext resulting from *repeated applications* of a computation, for verifiable computation the main focus is to minimize the client's computational effort within a *single* computation.

1.4 Paper Organization

The remainder of this paper is organized as follows. In Section 2 we present the basic tools that are used in our constructions. In Section 3 we formalize the notion of targeted malleability. In Sections 4 and 5 we present our constructions. Finally, in Section 6 we discuss possible extensions of our work and several open problems.

2 Preliminaries

In this section we present the basic tools that are used in our constructions: public-key encryption and homomorphic encryption, succinct non-interactive arguments, and non-interactive zero-knowledge proofs.

2.1 Public-Key Encryption

A public-key encryption scheme is a triplet $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ of probabilistic polynomial-time algorithms, where KeyGen is the key-generation algorithm, Enc is the encryption algorithm, and Dec is the decryption algorithm. The key-generation algorithm KeyGen receives as input the security parameter, and outputs a public key pk and a secret key sk. The encryption algorithm Enc receives as input a public key pk and a message m, and outputs a ciphertext c. The decryption algorithm Dec receives as input a ciphertext c and a secret key sk, and outputs a message m or the symbol $\perp$.

Functionality. In terms of functionality, in this paper we require the property of *almost-all-keys perfect decryption* [DNR04], defined as follows:

Definition 2.1. *A public-key encryption scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *has* almost-all-keys perfect decryption *if there exists a negligible function* $\nu(k)$ *such that for all sufficiently large* k *with probability* $1 - \nu(k)$ *over the choice of* $(sk, pk) \leftarrow \mathsf{KeyGen}(1^k)$*, for any message* m *it holds that*

$$\Pr\left[\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m\right] = 1 \ ,$$

where the probability is taken over the internal randomness of Enc *and* Dec*.*

We note that Dwork, Naor, and Reingold [DNR04] proposed a general transformation turning any encryption scheme into one that has almost-all-keys perfect decryption. When starting with a scheme that has a very low error probability, their transformation only changes the random bits used by the encryption algorithm, and in our setting this is important as it preserves the homomorphic operations. When starting with a scheme that has a significant error probability, we note that the error probability can be reduced exponentially by encrypting messages under several independently chosen public keys, and decrypting according to the majority. This again preserves the homomorphic operations.

Security. In terms of security, we consider the most basic notion of semantic-security against chosen-plaintext attacks, asking that any efficient adversary has only a negligible advantage in distinguishing between encryptions of different messages. This is formalized as follows:

Definition 2.2. *A public-key encryption scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is* semantically secure against chosen-plaintext attacks *if for any probabilistic polynomial-time adversary* $A = (A_1, A_2)$ *it holds that*

$$\mathsf{Adv}^{\mathsf{CPA}}_{\Pi,A}(k) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Expt}^{\mathsf{CPA}}_{\Pi,A,0}(k) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{CPA}}_{\Pi,A,1}(k) = 1\right]\right|$$

is negligible in k, where $\mathsf{Expt}^{\mathsf{CPA}}_{\Pi,A,b}(k)$ is defined as follows:

1. $(sk, pk) \leftarrow \mathsf{KeyGen}(1^k)$.
2. $(m_0, m_1, \mathsf{state}) \leftarrow A_1(1^k, pk)$ *such that* $|m_0| = |m_1|$.
3. $c^* \leftarrow \mathsf{Enc}_{pk}(m_b)$.
4. $b' \leftarrow A_2(c^*, \mathsf{state})$
5. *Output* b'.

Homomorphic encryption. A public-key encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is *homomorphic* with respect to a set of efficiently computable functions $\mathcal{F}$ if there exists a *homomorphic evaluation* algorithm $\mathsf{HomEval}$ that receives as input a public key pk, an encryption of a message m, and a function $f \in \mathcal{F}$, and outputs an encryption of the message $f(m)$. Formally, with overwhelming probability over the choice of $(sk, pk) \leftarrow \mathsf{KeyGen}(1^k)$ (as in Definition 2.1), for any ciphertext c such that $\mathsf{Dec}_{sk}(c) \neq \perp$ and for any function $f \in \mathcal{F}$ it holds that $\mathsf{Dec}_{sk}\left(\mathsf{HomEval}_{pk}(c, f)\right) = f\left(\mathsf{Dec}_{sk}(c)\right)$ with probability 1 over the internal randomness of $\mathsf{HomEval}$ and Dec.

The main property that is typically required from a homomorphic encryption scheme is *compactness*, asking that the length of the ciphertext does not trivially grow with the number of repeated homomorphic operations. In our setting, given an upper bound t on the number of repeated homomorphic operations that can be applied to a ciphertext produced by the encryption algorithm, we are interested in minimizing the dependency of the length of the ciphertext on t.

An additional property, that we do not consider in this paper, is of *function privacy*. Informally, this property asks that the homomorphic evaluation algorithm does not reveal which function from the set $\mathcal{F}$ it receives as input. We refer the reader to [GHV10] for a formal definition. We note that in our setting, where function privacy is not taken into account, we can assume without loss of generality that the homomorphic evaluation algorithm is deterministic.

2.2 Non-Interactive Extractable Arguments

A non-interactive argument system for a language $L = \bigcup_{k\in\mathbb{N}} L(k)$ with a witness relation $R = \bigcup_{k\in\mathbb{N}} R(k)$ consists of a triplet of algorithms $(\mathsf{CRSGen}, \mathsf{P}, \mathsf{V})$, where CRSGen is an algorithm generating a common reference string crs, and P and V are the prover and verifier algorithms, respectively. The prover takes as input a triplet (x, w, crs), where $(x, w) \in R$, and outputs an argument π. The verifier takes as input a triplet (x, π, crs) and either accepts or rejects. In this paper we consider a setting where all three algorithms run in polynomial time, CRSGen and P may be probabilistic, and V is deterministic.

We require three properties from such a system. The first property is *perfect completeness*: for every (x, w, crs) such that $(x, w) \in R$, the prover always generates an argument that is accepted by the verifier. The second property is *knowledge extraction*: for every efficient malicious prover P^* there exists an efficient "knowledge extractor" $\mathsf{Ext}_{\mathsf{P}^*}$, such that whenever P^* outputs (x, π) that is accepted by the verifier, $\mathsf{Ext}_{\mathsf{P}^*}$ when given the random coins of P^* can in fact produce a witness w such that $(x, w) \in R$ with all but a negligible probability. We note that this implies, in particular, soundness against efficient provers.

The perfect completeness and knowledge extraction properties are in fact trivial to satisfy: the prover can output the witness w as an argument, and the verifier checks that $(x, w) \in R$ (unlike for CS proofs [Mic00, Val08] we do not impose any non-trivial efficiency requirement on the verifier). The third property that we require from the argument system is that of having rather succinct

arguments: there should exist a constant $0 < \gamma < 1$ such that the arguments are of length at most $\gamma|w|$.

Definition 2.3. *Let $0 < \gamma < 1$ be a constant. A* γ-succinct non-interactive extractable argument system *for a language $L = \bigcup_{k\in\mathbb{N}} L(k)$ with a witness relation $R_L = \bigcup_{k\in\mathbb{N}} R_{L(k)}$ is a triplet of probabilistic polynomial-time algorithms $(\mathsf{CRSGen}, \mathsf{P}, \mathsf{V})$ with the following properties:*

1. **Perfect completeness:** *For every $k \in \mathbb{N}$ and $(x, w) \in R_{L(k)}$ it holds that*

$$\Pr\left[\mathsf{V}(1^k, x, \pi, \mathsf{crs}) = 1 \;\middle|\; \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k) \\ \pi \leftarrow \mathsf{P}(1^k, x, w, \mathsf{crs}) \end{array}\right] = 1$$

   where the probability is taken over the internal randomness of CRSGen, P *and* V.

2. **Adaptive knowledge extraction:** *For every probabilistic polynomial-time algorithm P^* there exist a probabilistic polynomial-time algorithm $\mathsf{Ext}_{\mathsf{P}^*}$ and a negligible function $\nu(\cdot)$ such that*

$$\Pr\left[(x, w) \notin R_{L(k)} \text{ and } \mathsf{V}(1^k, x, \pi, \mathsf{crs}) = 1 \;\middle|\; \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k), r \leftarrow \{0,1\}^* \\ (x, \pi) \leftarrow \mathsf{P}^*(1^k, \mathsf{crs}; r) \\ w \leftarrow \mathsf{Ext}_{\mathsf{P}^*}(1^k, \mathsf{crs}, r) \end{array}\right] \leq \nu(k)$$

   for all sufficiently large k, where the probability is taken over the internal randomness of CRSGen, P^*, V, *and* $\mathsf{Ext}_{\mathsf{P}^*}$.

3. **γ-Succinct arguments:** *For every $k \in \mathbb{N}$, $(x, w) \in R_{L(k)}$ and $\mathsf{crs} \in \{0,1\}^*$, it holds that $\mathsf{P}(1^k, x, w, \mathsf{crs})$ produces a distribution over strings of length at most $\gamma|w|$.*

Instantiation. An argument system satisfying Definition 2.3 (with a deterministic verifier) was recently constructed by Groth [Gro10] in the common-reference string model based on a certain "knowledge of exponent" assumption. His scheme is even zero-knowledge, and the length of the resulting arguments is essentially independent of the length of the witness. The length of the common-reference string, however, is at least quadratic in the length of the witness[4], and this will limit our constructions to support only a constant number of repeated homomorphic operations. Any argument system satisfying Definition 2.3 with a common-reference string of length linear in the length of the witness will allow our first construction to support any logarithmic number of repeated homomorphic operations, and our second construction to support any polynomial number of such operations.

The running time of the knowledge extractor. The proofs of security of our constructions involve nested invocations of the knowledge extractors that are provided by Definition 2.3. When supporting only a constant number of repeated homomorphic operations the simulation will always run in polynomial time. When supporting a super-constant number of repeated homomorphic operations, we need to require that the knowledge extractor $\mathsf{Ext}_{\mathsf{P}^*}$ corresponding to a malicious prover P^* runs in time that is linear in the running time of P^*. This (together with a common-reference string of linear length) will allow our first construction to support any logarithmic number of repeated homomorphic operations, and our second construction to support any polynomial number of such operations.

[4]For proving the satisfiability of a circuit of size s, the common-reference string in [Gro10] consists of $O(s^2)$ group elements, taken from a group where (in particular) the discrete logarithm problem is assumed to be hard. Lipmaa [Lip11] was able to slightly reduce the number of group elements, but even in his construction it is still super-linear.

2.3 Non-Interactive Simulation-Sound Adaptive Zero-Knowledge Proofs

We define the notion of a non-interactive simulation-sound adaptive zero-knowledge proof system [BFM88, FLS90, BSM+91, Sah99].

Definition 2.4. *A* non-interactive simulation-sound adaptive zero-knowledge proof system *for a language* $L = \bigcup_{k\in\mathbb{N}} L(k)$ *with a witness relation* $R_L = \bigcup_{k\in\mathbb{N}} R_{L(k)}$ *is a tuple of probabilistic polynomial-time algorithms* $\Pi = (\mathsf{CRSGen}, \mathsf{P}, \mathsf{V}, \mathsf{S}_1, \mathsf{S}_2)$ *with the following properties:*

1. **Perfect completeness:** *For every* $k \in \mathbb{N}$ *and* $(x, w) \in R_{L(k)}$ *it holds that*

$$\Pr\left[\mathsf{V}(1^k, x, \pi, \mathsf{crs}) = 1 \;\middle|\; \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k) \\ \pi \leftarrow \mathsf{P}(1^k, x, w, \mathsf{crs}) \end{array}\right] = 1$$

   where the probability is taken over the internal randomness of CRSGen, P *and* V.

2. **Adaptive soundness:** *For every algorithm* P^* *there exists a negligible function* $\nu(\cdot)$ *such that*

$$\Pr\left[x \notin L(k) \text{ and } \mathsf{V}(1^k, x, \pi, \mathsf{crs}) = 1 \;\middle|\; \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k) \\ (x, \pi) \leftarrow \mathsf{P}^*(1^k, \mathsf{crs}) \end{array}\right] \leq \nu(k)$$

   for all sufficiently large k*, where the probability is taken over the internal randomness of* CRSGen, P^*, *and* V.

3. **Adaptive zero knowledge:** *For every probabilistic polynomial-time algorithm* A *there exists a negligible function* $\nu(\cdot)$ *such that*

$$\mathsf{Adv}^{\mathsf{ZK}}_{\Pi,A}(k) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Expt}^{\mathsf{ZK}}_{\Pi,A}(k) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{ZK}}_{\Pi,A,\mathsf{S}_1,\mathsf{S}_2}(k) = 1\right]\right| \leq \nu(k)$$

   for all sufficiently large k*, where the experiment* $\mathsf{Expt}^{\mathsf{ZK}}_{\Pi,A}(k)$ *is defined as:*

   (a) $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k)$
   (b) $b \leftarrow A^{\mathsf{P}(1^k,\cdot,\cdot,\mathsf{crs})}(1^k, \mathsf{crs})$
   (c) Output b

   and the experiment $\mathsf{Expt}^{\mathsf{ZK}}_{\Pi,A,\mathsf{S}_1,\mathsf{S}_2}(k)$ *is defined as:*

   (a) $(\mathsf{crs}, \tau) \leftarrow \mathsf{S}_1(1^k)$
   (b) $b \leftarrow A^{\mathsf{S}'_2(1^k,\cdot,\cdot,\tau)}(1^k, \mathsf{crs})$*, where* $\mathsf{S}'_2(1^k, x, w, \tau) = \mathsf{S}_2(1^k, x, \tau)$
   (c) output b

4. **Simulation soundness:** *For every probabilistic polynomial-time algorithm* A *there exists a negligible function* $\nu(\cdot)$ *such that*

$$\mathsf{Adv}^{\mathsf{SS}}_{\Pi,A}(k) \stackrel{\mathsf{def}}{=} \Pr\left[\mathsf{Expt}^{\mathsf{SS}}_{\Pi,A}(k) = 1\right] \leq \nu(k)$$

   for all sufficiently large k*, where the experiment* $\mathsf{Expt}^{\mathsf{SS}}_{\Pi,A}(k)$ *is defined as:*

   (a) $(\mathsf{crs}, \tau) \leftarrow \mathsf{S}_1(1^k)$
   (b) $(x, \pi) \leftarrow A^{\mathsf{S}_2(1^k,\cdot,\tau)}(1^k, \mathsf{crs})$
   (c) Denote by Q *the set of* S_2*'s answers to* A*'s oracle queries*
   (d) Output 1 *if and only if* $x \notin L(k)$, $\pi \notin Q$, *and* $\mathsf{V}(1^k, x, \pi, \mathsf{crs}) = 1$

3 Defining Targeted Malleability

In this section we introduce a framework for targeted malleability by formalizing *non-malleability with respect to a specific set of functions*. We begin by discussing the case of *univariate* functions, and then show that our approach naturally generalizes to the case of *multivariate* functions. Given an encryption scheme that is homomorphic with respect to a set of functions $\mathcal{F}$ we would like to capture the following notion of security: For any efficient adversary that is given an encryption c of a message m and outputs an encryption c' of a message m', it should hold that either (1) m' is independent of m, (2) $c' = c$ (and thus $m' = m$), or (3) c' is obtained by repeatedly applying the homomorphic evaluation algorithm on c using functions $f_1, \ldots, f_\ell \in \mathcal{F}$. The first two properties are the standard ones for non-malleability [DDN00], and the third property captures targeted malleability.

Following [DDN00, BS99, PSV07] we formalize a simulation-based notion of security that compares a real-world adversary to a simulator that is not given any ciphertexts as input. Specifically, we consider two experiments: a real-world experiment, and a simulated experiment, and require that for any efficient real-world adversary there exists an efficient simulator such that the outputs of the two experiments are computationally indistinguishable[5]. We consider both chosen-plaintext attacks (CPA) and a-priori chosen-ciphertext attacks (CCA1). We assume that the set of functions $\mathcal{F}$ is recognizable in polynomial time, and it may or may not be closed under composition.

Chosen-plaintext attacks (CPA). In the real-world experiment we consider adversaries that are described by two algorithms $A = (A_1, A_2)$. The algorithm A_1 takes as input the public key of the scheme, and outputs a description of a distribution $\mathcal{M}$ over messages, a state information state_1 to be included in the output of the experiment, and a state information state_2 to be given as input to the algorithm A_2. We note that state_1 and state_2 may include pk and $\mathcal{M}$. Then, the algorithm A_2 takes as input the state information state_2 and a sequence of ciphertexts that are encryptions of messages $m_1, \ldots, m_r$ sampled from $\mathcal{M}$. The algorithm A_2 outputs a sequence of ciphertexts $c_1, \ldots, c_q$, and the output of the experiment is defined as $(\mathsf{state}_1, m_1, \ldots, m_r, d_1, \ldots, d_q)$, where for every $j \in \{1, \ldots, q\}$ the value d_j is one of two things: if c_j is equal to the i-th input ciphertext for some i then d_j is a special symbol copy_i; otherwise d_j is the decryption of c_j.

In the simulated experiment the simulator is also described by two algorithms $S = (S_1, S_2)$. The algorithm S_1 takes as input the public key, and outputs a description of a distribution $\mathcal{M}$ over messages, a state information state_1 to be included in the output of the experiment, and a state information state_2 to be given as input to the algorithm S_2 (as in the real world). Then, a sequence of messages is sampled from $\mathcal{M}$, but here the algorithm S_2 does not receive the encryptions of these messages, but only state_2. The algorithm S_2 should output q values, where each value can take one of three possible types. The first type is the special symbol copy_i, and in this case we define $d_j = \mathsf{copy}_i$. This captures the ability of real-world adversary to copy one of the ciphertexts. The second type is an index $i \in \{1, \ldots, r\}$ and a sequence of functions $f_1, \ldots, f_\ell \in \mathcal{F}$, where ℓ is at most some predetermined upper bound t on the number of repeated homomorphic operations. In this case we define $d_j = f(m_i)$ where $f = f_1 \circ \cdots \circ f_\ell$. This captures the ability of the real-world adversary to choose one of its input ciphertexts and apply the homomorphic evaluation algorithm for at most t times. The third type is a ciphertext c_j, and in this case d_j is defined as its decryption. As the simulator does not receive any ciphertexts as input, this captures the ability of the adversary to produce a ciphertext that is independent of its input ciphertexts. The output of the experiment

[5] As commented by Pass et al. [PSV07], note that a distinguisher between the two experiments corresponds to using a relation for capturing non-malleability as in [DDN00, BS99].

is defined as $(\mathsf{state}_1, m_1, \ldots, m_r, d_1, \ldots, d_q)$.

Definition 3.1. *Let $t = t(k)$ be a polynomial. A public-key encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{HomEval})$ is t-bounded non-malleable against chosen-plaintext attacks with respect to a set of functions $\mathcal{F}$ if for any polynomials $r = r(k)$ and $q = q(k)$ and for any probabilistic polynomial-time algorithm $A = (A_1, A_2)$ there exists a probabilistic polynomial-time algorithm $S = (S_1, S_2)$ such that the distributions $\left\{\mathsf{Real}^{\mathsf{CPA}}_{\Pi,A,t,r,q}(k)\right\}_{k\in\mathbb{N}}$ and $\left\{\mathsf{Sim}^{\mathsf{CPA}}_{\Pi,S,t,r,q}(k)\right\}_{k\in\mathbb{N}}$ (see Figure 1) are computationally indistinguishable.*

$\mathsf{Real}^{\mathsf{CPA}}_{\Pi,A,t,r,q}(k)$:

1. $(sk, pk) \leftarrow \mathsf{KeyGen}(1^k)$
2. $(\mathcal{M}, \mathsf{state}_1, \mathsf{state}_2) \leftarrow A_1(1^k, pk)$
3. $(m_1, \ldots, m_r) \leftarrow \mathcal{M}$
4. $c_i^* \leftarrow \mathsf{Enc}_{pk}(m_i)$ for every $i \in \{1, \ldots, r\}$
5. $(c_1, \ldots, c_q) \leftarrow A_2(1^k, c_1^*, \ldots, c_r^*, \mathsf{state}_2)$
6. For every $j \in \{1, \ldots, q\}$ let
$$d_j = \begin{cases} \mathsf{copy}_i & \text{if } c_j = c_i^* \\ \mathsf{Dec}_{sk}(c_j) & \text{otherwise} \end{cases}$$
7. Output $(\mathsf{state}_1, m_1, \ldots, m_r, d_1, \ldots, d_q)$

$\mathsf{Sim}^{\mathsf{CPA}}_{\Pi,S,t,r,q}(k)$:

1. $(sk, pk) \leftarrow \mathsf{KeyGen}(1^k)$
2. $(\mathcal{M}, \mathsf{state}_1, \mathsf{state}_2) \leftarrow S_1(1^k, pk)$
3. $(m_1, \ldots, m_r) \leftarrow \mathcal{M}$
4. $(c_1, \ldots, c_q) \leftarrow S_2(1^k, \mathsf{state}_2)$
5. For every $j \in \{1, \ldots, q\}$ let
$$d_j = \begin{cases} \mathsf{copy}_i & \text{if } c_j = \mathsf{copy}_i \\ f(m_i) & \text{if } c_j = (i, f_1, \ldots, f_\ell) \text{ where } i \in \{1, \ldots, r\}, \\ & \ell \le t, f_1, \ldots, f_\ell \in \mathcal{F}, \\ & \text{and } f = f_1 \circ \cdots \circ f_\ell \\ \mathsf{Dec}_{sk}(c_j) & \text{otherwise} \end{cases}$$
6. Output $(\mathsf{state}_1, m_1, \ldots, m_r, d_1, \ldots, d_q)$

Figure 1: The distributions $\mathsf{Real}^{\mathsf{CPA}}_{\Pi,A,t,r,q}(k)$ and $\mathsf{Sim}^{\mathsf{CPA}}_{\Pi,S,t,r,q}(k)$.

Dealing with multivariate functions. Our approach naturally generalizes to the case of multivariate functions as follows. Fix a set $\mathcal{F}$ of functions that are defined on d-tuples of plaintexts for some integer d, and let A be an efficient adversary that is given a sequence of ciphertexts $c_1^*, \ldots, c_r^*$ and outputs a sequence of ciphertexts $c_1, \ldots, c_q$, as in Definition 3.1. Intuitively, for each output ciphertext c_j it should hold that either (1) $\mathsf{Dec}_{sk}(c_j)$ is independent of $c_1^*, \ldots, c_r^*$, (2) $c_j = c_i^*$ for some $i \in \{1, \ldots, r\}$, or (3) c_j is obtained by repeatedly applying the homomorphic evaluation algorithm using functions from the set $\mathcal{F}$ and a sequence of ciphertexts where each ciphertext is either taken from $c_1^*, \ldots, c_r^*$ or is independent of $c_1^*, \ldots, c_r^*$.

Formally, the distribution $\mathsf{Real}^{\mathsf{CPA}}_{\Pi,A,t,r,q}(k)$ is not modified, and the distribution $\mathsf{Sim}^{\mathsf{CPA}}_{\Pi,S,t,r,q}(k)$ is modified by only changing the output $c_j = (i, f_1, \ldots, f_\ell)$ of S_2 to a d-ary tree of depth at most t: each internal node contains a description of a function from the set $\mathcal{F}$, and each leaf contains either an index $i \in \{1, \ldots, r\}$ or a plaintext m. The corresponding value d_j is then computed by evaluating the tree bottom-up where each index i is replaced by the plaintext m_i that was sampled from $\mathcal{M}$.

Dealing with randomized functions. The above definitions assume that $\mathcal{F}$ is a set of *deterministic* functions. More generally, one can also consider *randomized* functions. There are two natural

approaches for extending our framework to this setting. The first approach is to view each function $f \in \mathcal{F}$ and string $r \in \{0,1\}^*$ (of an appropriate length) as defining a function $f_r(m) = f(m;r)$, and to apply the above definitions to the set $\mathcal{F}' = \{f_r\}_{f\in\mathcal{F},r\in\{0,1\}^*}$ of deterministic functions. The second approach is to modify the distribution $\mathsf{Sim}^{\mathsf{CPA}}_{\Pi,S,t,r,q}(k)$ as follows: instead of setting d_j to the value $f(m_i)$, we sample d_j from the distribution induced by the random variable $f(m_i)$. Each of these two approaches may be preferable depending on the context in which the encryption scheme is used, and for simplifying the presentation in this paper we assume that $\mathcal{F}$ is a set of deterministic functions.

A-priori chosen-ciphertexts attacks (CCA1). Definition 3.1 generalizes to a-priori chosen-ciphertext attacks by providing the algorithm A_1 oracle access to the decryption oracle before choosing the distribution $\mathcal{M}$. At the same time, however, the simulator still needs to specify the distribution $\mathcal{M}$ without having such access (this is also known as *non-assisted simulation*). Specifically, we define $\{\mathsf{Real}^{\mathsf{CCA1}}_{\Pi,A,t,r,q}(k)\}_{k\in\mathbb{N}}$ and $\{\mathsf{Sim}^{\mathsf{CCA1}}_{\Pi,S,t,r,q}(k)\}_{k\in\mathbb{N}}$ as follows: $\mathsf{Real}^{\mathsf{CCA1}}_{\Pi,A,t,r,q}(k)$ is obtained from $\mathsf{Real}^{\mathsf{CPA}}_{\Pi,A,t,r,q}(k)$ by providing A_1 with oracle access to $\mathsf{Dec}_{sk}(\cdot)$, and $\mathsf{Sim}^{\mathsf{CCA1}}_{\Pi,S,t,r,q}(k)$ is identical to $\mathsf{Sim}^{\mathsf{CPA}}_{\Pi,S,t,r,q}(k)$.

Definition 3.2. *Let $t = t(k)$ be a polynomial. A public-key encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{HomEval})$ is* t-bounded non-malleable against a-priori chosen-ciphertext attacks with respect to a set of functions $\mathcal{F}$ *if for any polynomials $r = r(k)$ and $q = q(k)$ and for any probabilistic polynomial-time algorithm $A = (A_1, A_2)$ there exists a probabilistic polynomial-time algorithm $S = (S_1, S_2)$ such that the distributions $\{\mathsf{Real}^{\mathsf{CCA1}}_{\Pi,A,t,r,q}(k)\}_{k\in\mathbb{N}}$ and $\{\mathsf{Sim}^{\mathsf{CCA1}}_{\Pi,S,t,r,q}(k)\}_{k\in\mathbb{N}}$ are computationally indistinguishable.*

4 The Path-Based Construction

In this section we present our first construction. The construction is based on any public-key encryption scheme that is homomorphic with respect to some set $\mathcal{F}$ of functions, a non-interactive zero-knowledge proof system, and γ-succinct non-interactive argument systems for $\gamma = 1/4$. The scheme is parameterized by an upper bound t on the number of repeated homomorphic operations that can be applied to a ciphertext produced by the encryption algorithm. The scheme enjoys the feature that the dependency on t is essentially eliminated from the length of the ciphertext, and shifted to the public key. The public key consists of $t + 1$ common reference strings: one for the zero-knowledge proof system, and t for the succinct argument systems. We note that in various cases (such as argument systems in the common *random* string model) it may be possible to use only one common-reference string for all t argument systems, and then the length of the public key decreases quite significantly.

In Section 4.1 we formally specify the building blocks of the scheme, and in Section 4.2 we provide a description of the scheme. In Section 4.3 we prove the security of the scheme against chosen-plaintexts attacks (CPA), and in Section 4.4 we show tat the proof in fact extends to deal with a-priori chosen-ciphertext attacks (CCA1).

4.1 The Building Blocks

Our construction relies on the following building blocks:

1. A homomorphic public-key encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{HomEval})$ with respect to an efficiently recognizable set of efficiently computable functions $\mathcal{F}$. We assume that the

scheme has almost-all-keys perfect decryption (see Definition 2.1). In addition, as discussed in Section 2.1, as we do not consider function privacy we assume without loss of generality that $\mathsf{HomEval}$ is deterministic.

For any security parameter $k \in \mathbb{N}$ we denote by $\ell_{pk} = \ell_{pk}(k)$, $\ell_m = \ell_m(k)$, $\ell_r = \ell_r(k)$, and $\ell_c = \ell_c(k)$ the bit-lengths of the public key, plaintext, randomness of Enc, and ciphertext[6], respectively, for the scheme Π. In addition, we denote by $\mathsf{V}_{\mathcal{F}}$ the deterministic polynomial-time algorithm for testing membership in the set $\mathcal{F}$, and denote by $\ell_{\mathcal{F}} = \ell_{\mathcal{F}}(k)$ the bit-length of the description of each function $f \in \mathcal{F}$.

2. A non-interactive deterministic-verifier simulation-sound adaptive zero-knowledge proof system (see Section 2.3) $\Pi^{(0)} = \left(\mathsf{CRSGen}^{(0)}, \mathsf{P}^{(0)}, \mathsf{V}^{(0)}\right)$ for the $\mathcal{NP}$-language $L^{(0)} = \bigcup_{k\in\mathbb{N}} L^{(0)}(k)$ defined as follows.

$$L^{(0)}(k) = \left\{ \left(pk_0, pk_1, c_0^{(0)}, c_1^{(0)}\right) \in \{0,1\}^{2\ell_{pk}+2\ell_c} : \begin{array}{c} \exists (m, r_0, r_1) \in \{0,1\}^{\ell_m+2\ell_r} \text{ s.t.} \\ c_0^{(0)} = \mathsf{Enc}_{pk_0}(m; r_0) \\ \text{and } c_1^{(0)} = \mathsf{Enc}_{pk_1}(m; r_1) \end{array} \right\}$$

For any security parameter $k \in \mathbb{N}$ we denote by $\ell_{\mathsf{crs}^{(0)}} = \ell_{\mathsf{crs}^{(0)}}(k)$ and $\ell_{\pi^{(0)}} = \ell_{\pi^{(0)}}(k)$ the bit-lengths of the common-reference strings produced by $\mathsf{CRSGen}^{(0)}$ and of the proofs produced by $\mathsf{P}^{(0)}$, respectively. Without loss of generality we assume that $\ell_{\pi^{(0)}} \geq \max\{\ell_c, \ell_{\mathcal{F}}\}$ (as otherwise proofs can always be padded).

3. For every $i \in \{1, \ldots, t\}$ a 1/4-succinct non-interactive deterministic-verifier extractable argument system (see Section 2.2) $\Pi^{(i)} = \left(\mathsf{CRSGen}^{(i)}, \mathsf{P}^{(i)}, \mathsf{V}^{(i)}\right)$ for the $\mathcal{NP}$-language $L^{(i)} = \bigcup_{k\in\mathbb{N}} L^{(i)}(k)$ defined as follows.

$$L^{(i)}(k) = \left\{ \begin{array}{l} \left(pk_0, pk_1, c_0^{(i)}, c_1^{(i)}, \mathsf{crs}^{(i-1)}, \ldots, \mathsf{crs}^{(0)}\right) \in \{0,1\}^{2\ell_{pk}+2\ell_c+\sum_{j=0}^{i-1}\ell_{\mathsf{crs}^{(j)}}} : \\ \exists \left(c_0^{(i-1)}, c_1^{(i-1)}, \pi^{(i-1)}, f\right) \in \{0,1\}^{2\ell_c+\ell_{\pi^{(i-1)}}+\ell_{\mathcal{F}}} \text{ s.t.} \\ \quad \bullet \ \mathsf{V}_{\mathcal{F}}(f) = 1 \\ \quad \bullet \ c_0^{(i)} = \mathsf{HomEval}_{pk_0}\left(c_0^{(i-1)}, f\right) \\ \quad \bullet \ c_1^{(i)} = \mathsf{HomEval}_{pk_1}\left(c_1^{(i-1)}, f\right) \\ \quad \bullet \ \mathsf{V}^{(i-1)}\left(\left(pk_0, pk_1, c_0^{(i-1)}, c_1^{(i-1)}, \mathsf{crs}^{(i-2)}, \ldots, \mathsf{crs}^{(0)}\right), \pi^{(i-1)}, \mathsf{crs}^{(i-1)}\right) = 1 \end{array} \right\}$$

For any security parameter $k \in \mathbb{N}$ we denote by $\ell_{\mathsf{crs}^{(i)}} = \ell_{\mathsf{crs}^{(i)}}(k)$ and $\ell_{\pi^{(i)}} = \ell_{\pi^{(i)}}(k)$ the bit-lengths of the common-reference strings produced by $\mathsf{CRSGen}^{(i)}$ and of the arguments produced by $\mathsf{P}^{(i)}$, respectively.

4.2 The Scheme

The scheme $\Pi' = (\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{HomEval}')$ is parameterized by an upper bound t on the number of repeated homomorphic operations that can be applied to a ciphertext produced by the encryption algorithm. The scheme is defined as follows:

[6]For simplicity we assume a fixed upper bound ℓ_c on the length of ciphertexts, but this is not essential to our construction. More generally, one can allow the length of ciphertexts to increase as a result of applying the homomorphic operation.

- **Key generation:** On input 1^k sample two pairs of keys $(sk_0, pk_0) \leftarrow \mathsf{KeyGen}(1^k)$ and $(sk_1, pk_1) \leftarrow \mathsf{KeyGen}(1^k)$. Then, for every $i \in \{0, \ldots, t\}$ sample $\mathsf{crs}^{(i)} \leftarrow \mathsf{CRSGen}^{(i)}(1^k)$. Output the secret key $sk = (sk_0, sk_1)$ and the public key $pk = \left(pk_0, pk_1, \mathsf{crs}^{(0)}, \ldots, \mathsf{crs}^{(t)}\right)$.

- **Encryption:** On input a public key pk and a plaintext m, sample $r_0, r_1 \in \{0,1\}^*$ uniformly at random, and output the ciphertext $c^{(0)} = \left(0, c_0^{(0)}, c_1^{(0)}, \pi^{(0)}\right)$, where

$$\begin{aligned} c_0^{(0)} &= \mathsf{Enc}_{pk_0}(m; r_0) \ , \\ c_1^{(0)} &= \mathsf{Enc}_{pk_1}(m; r_1) \ , \\ \pi^{(0)} &\leftarrow \mathsf{P}^{(0)}\left(\left(pk_0, pk_1, c_0^{(0)}, c_1^{(0)}\right), (m, r_0, r_1), \mathsf{crs}^{(0)}\right) \ . \end{aligned}$$

- **Homomorphic evaluation:** On input a public key pk, a ciphertext $\left(i, c_0^{(i)}, c_1^{(i)}, \pi^{(i)}\right)$, and a function $f \in \mathcal{F}$, proceed as follows. If $i \notin \{0, \ldots, t-1\}$ or

$$\mathsf{V}^{(i)}\left(\left(pk_0, pk_1, c_0^{(i)}, c_1^{(i)}, \mathsf{crs}^{(i-1)}, \ldots, \mathsf{crs}^{(0)}\right), \pi^{(i)}, \mathsf{crs}^{(i)}\right) = 0$$

  then output $\perp$. Otherwise, output the ciphertext $c^{(i+1)} = \left(i+1, c_0^{(i+1)}, c_1^{(i+1)}, \pi^{(i+1)}\right)$, where

$$\begin{aligned} c_0^{(i+1)} &= \mathsf{HomEval}_{pk_0}\left(c_0^{(i)}, f\right) \ , \\ c_1^{(i+1)} &= \mathsf{HomEval}_{pk_1}\left(c_1^{(i)}, f\right) \ , \\ \pi^{(i+1)} &\leftarrow \mathsf{P}^{(i+1)}\left(\left(pk_0, pk_1, c_0^{(i+1)}, c_1^{(i+1)}, \mathsf{crs}^{(i)}, \ldots, \mathsf{crs}^{(0)}\right), \left(c_0^{(i)}, c_1^{(i)}, \pi^{(i)}, f\right), \mathsf{crs}^{(i+1)}\right) \ . \end{aligned}$$

- **Decryption:** On input a secret key sk and a ciphertext $\left(i, c_0^{(i)}, c_1^{(i)}, \pi^{(i)}\right)$, output $\perp$ if $i \notin \{0, \ldots, t\}$ or

$$\mathsf{V}^{(i)}\left(\left(pk_0, pk_1, c_0^{(i)}, c_1^{(i)}, \mathsf{crs}^{(i-1)}, \ldots, \mathsf{crs}^{(0)}\right), \pi^{(i)}, \mathsf{crs}^{(i)}\right) = 0 \ .$$

  Otherwise, compute $m_0 = \mathsf{Dec}_{sk_0}\left(c_0^{(i)}\right)$ and $m_1 = \mathsf{Dec}_{sk_1}\left(c_1^{(i)}\right)$. If $m_0 \neq m_1$ then output $\perp$, and otherwise output m_0.

Note that at any point in time a ciphertext of the scheme is of the form $\left(i, c_0^{(i)}, c_1^{(i)}, \pi^{(i)}\right)$, where $i \in \{0, \ldots, t\}$, $c_0^{(i)}$ and $c_1^{(i)}$ are ciphertexts of the underlying encryption scheme, and $\pi^{(i)}$ is a proof or an argument with respect to one of $\Pi^{(0)}, \ldots, \Pi^{(t)}$. Note that the assumption that the argument systems $\Pi^{(1)}, \ldots, \Pi^{(t)}$ are 1/4-succinct implies that the length of their arguments is upper bounded by length of the proofs of $\Pi^{(0)}$ (i.e., $\ell_{\pi^{(i)}} \leq \ell_{\pi^{(0)}}$ for every $i \in \{1, \ldots, t\}$). Thus, the only dependency on t in the length of the ciphertext results from the $\lceil \log_2(t+1) \rceil$ bits describing the prefix i.

4.3 Chosen-Plaintext Security

We now prove that the construction offers targeted malleability against chosen-plaintext attacks. For concreteness we focus on the case of a single message and a single ciphertext (i.e., the case $r(k) = q(k) = 1$ in Definition 3.1), and note that the more general case is a straightforward generalization. Given an adversary $A = (A_1, A_2)$ we construct a simulator $S = (S_1, S_2)$ as follows.

The algorithm S_1. The algorithm S_1 is identical to A_1, except for also including the public key pk and the distribution $\mathcal{M}$ in the state that it forwards to S_2. That is, S_1 on input $(1^k, pk)$ invokes A_1 on the same input to obtain a triplet $(\mathcal{M}, \mathsf{state}_1, \mathsf{state}_2)$, and then outputs $(\mathcal{M}, \mathsf{state}_1, \mathsf{state}'_2)$ where $\mathsf{state}'_2 = (pk, \mathcal{M}, \mathsf{state}_2)$.

The algorithm S_2. The algorithm S_2 on input $(1^k, \mathsf{state}'_2)$ where $\mathsf{state}'_2 = (pk, \mathcal{M}, \mathsf{state}_2)$, first samples $m' \leftarrow \mathcal{M}$, and computes $c^* \leftarrow \mathsf{Enc}'_{pk}(m')$. Then, it samples $r \leftarrow \{0,1\}^*$, and computes $c = \left(i, c_0^{(i)}, c_1^{(i)}, \pi^{(i)}\right) = A_2(1^k, c^*, \mathsf{state}_2; r)$. If $i \notin \{0, \ldots, t\}$ or

$$\mathsf{V}^{(i)}\left(\left(pk_0, pk_1, c_0^{(i)}, c_1^{(i)}, \mathsf{crs}^{(i-1)}, \ldots, \mathsf{crs}^{(0)}\right), \pi^{(i)}, \mathsf{crs}^{(i)}\right) = 0$$

then S_2 outputs c. Otherwise, S_2 utilizes the knowledge extractors guaranteed by the argument systems $\Pi^{(1)}, \ldots, \Pi^{(t)}$ to generate a "certification chain" for c of the form

$$\left(c^{(0)}, f^{(0)}, \ldots, c^{(i-1)}, f^{(i-1)}, c^{(i)}\right)$$

satisfying the following two properties:

1. $c^{(i)} = c$.
2. For every $j \in \{1, \ldots, i\}$ it holds that $c^{(j)} = \mathsf{HomEval}'_{pk}\left(c^{(j-1)}, f^{(j-1)}\right)$.

We elaborate below on the process of generating the certification chain. If S_2 fails in generating such a chain then it outputs c. Otherwise, S_2 computes its output as follows:

1. If $c^{(0)} = c^*$ and $i = 0$, then S_2 outputs copy_1.
2. If $c^{(0)} = c^*$ and $i > 0$, then S_2 outputs $f^{(0)} \circ \cdots \circ f^{(i-1)}$.
3. If $c^{(0)} \neq c^*$, then S_2 outputs c.

Generating the certification chain. We say that a ciphertext $\left(i, c_0^{(i)}, c_1^{(i)}, \pi^{(i)}\right)$ is *valid* if $i \in \{0, \ldots, t\}$ and

$$\mathsf{V}^{(i)}\left(\left(pk_0, pk_1, c_0^{(i)}, c_1^{(i)}, \mathsf{crs}^{(i-1)}, \ldots, \mathsf{crs}^{(0)}\right), \pi^{(i)}, \mathsf{crs}^{(i)}\right) = 1 \ .$$

Viewing the algorithm A_2 as a malicious prover with respect to the argument system $\Pi^{(i)}$ with the common reference string $\mathsf{crs}^{(i)}$, whenever A_2 outputs a valid ciphertext $c^{(i)} = \left(i, c_0^{(i)}, c_1^{(i)}, \pi^{(i)}\right)$, the algorithm S_2 invokes the knowledge extractor Ext_{A_2} that corresponds to A_2 (recall Definition 2.3) to obtain a witness $\left(c_0^{(i-1)}, c_1^{(i-1)}, \pi^{(i-1)}, f^{(i-1)}\right)$ to the fact that

$$\left(pk_0, pk_1, c_0^{(i)}, c_1^{(i)}, \mathsf{crs}^{(i-1)}, \ldots, \mathsf{crs}^{(0)}\right) \in L^{(i)} \ .$$

Note that S_2 chooses the randomness for A_2, which it can then provide to Ext_{A_2}. If successful then by the definition of $L^{(i)}$ we have a new valid cipertext $c^{(i-1)} = \left(i-1, c_0^{(i-1)}, c_1^{(i-1)}, \pi^{(i-1)}\right)$. If $i = 1$ then we are done. Otherwise (i.e., if $i > 1$), viewing the combination of A_2 and Ext_{A_2} as a malicious prover with respect to the argument system $\Pi^{(i-1)}$ with the common reference string $\mathsf{crs}^{(i-1)}$, the

algorithm S_2 invokes the knowledge extractor $\mathsf{Ext}_{(A_2,\mathsf{Ext}_{A_2})}$ that corresponds to the combination of A_2 and Ext_{A_2} to obtain a witness $\left(c_0^{(i-2)}, c_1^{(i-2)}, \pi^{(i-2)}, f^{(i-2)}\right)$ to the fact that

$$\left(pk_0, pk_1, c_0^{(i-1)}, c_1^{(i-1)}, \mathsf{crs}^{(i-2)}, \ldots, \mathsf{crs}^{(0)}\right) \in L^{(i-1)} \ ,$$

and so on for i iterations or until the first failure.

Having described the simulator we now prove the following theorem stating the security of the scheme in the case $r(k) = q(k) = 1$ (noting again that the more general case is a straightforward generalization). As discussed in Section 2.2, the quadratic blow-up in the length of the common-reference string in Groth's argument system [Gro10] restricts our treatment here to a constant number t of repeated homomorphic operations, and any improvement to Groth's argument system with a common-reference string of linear length will directly allow any logarithmic number of repeated homomorphic operations (and any polynomial number of such operations in the scheme presented in Section 5).

Theorem 4.1. *For any constant $t \in \mathbb{N}$ and for any probabilistic polynomial-time adversary A the distributions $\{\mathsf{Real}^{\mathsf{CPA}}_{\Pi',A,t,r,q}(k)\}_{k\in\mathbb{N}}$ and $\{\mathsf{Sim}^{\mathsf{CPA}}_{\Pi',S,t,r,q}(k)\}_{k\in\mathbb{N}}$ are computationally indistinguishable, for $r(k) = q(k) = 1$.*

Proof. We define a sequence of distributions $\mathcal{D}_1, \ldots, \mathcal{D}_7$ such that $\mathcal{D}_1 = \mathsf{Sim}^{\mathsf{CPA}}_{\Pi',S,t,r,q}$ and $\mathcal{D}_7 = \mathsf{Real}^{\mathsf{CPA}}_{\Pi',A,t,r,q}$, and prove that for every $i \in \{1, \ldots, 6\}$ the distributions $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$ are computationally indistinguishable. For simplicity in what follows we assume that the scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{HomEval})$ actually has perfect decryption for all keys (and not with an overwhelming probability over the choice of keys). This assumption clearly does not hurt any of the indistinguishability arguments in our proof, since we can initially condition on the event that both (sk_0, pk_0) and (sk_1, pk_1) provide perfect decryption.

The distribution $\mathcal{D}_1$. This is the distribution $\mathsf{Sim}^{\mathsf{CPA}}_{\Pi',S,t,r,q}$.

The distribution $\mathcal{D}_2$. This distribution is obtained from $\mathcal{D}_1$ via the following modification. As in $\mathcal{D}_1$, if S_2 fails to obtain a certification chain, then output $(\mathsf{state}_1, m, \perp)$. Otherwise, the output is computed as follows:

1. If $c^{(0)} = c^*$ and $i = 0$ then output $(\mathsf{state}_1, m, \mathsf{copy}_1)$. This is identical to $\mathcal{D}_1$.
2. If $c^{(0)} = c^*$ and $i > 0$ then output $(\mathsf{state}_1, m, f(m))$, where $f = f^{(0)} \circ \cdots \circ f^{(i-1)}$. This is identical to $\mathcal{D}_1$.
3. If $c^{(0)} \neq c^*$ then compute the message $m^{(0)} = \mathsf{Dec}'_{sk}(c^{(0)})$. If $m^{(0)} \neq \perp$ then output $\left(\mathsf{state}_1, m, f\left(m^{(0)}\right)\right)$, where $f = f^{(0)} \circ \cdots \circ f^{(i-1)}$, and otherwise output $(\mathsf{state}_1, m, \perp)$. That is, in this case instead of invoking the decryption algorithm Dec' on $c^{(i)}$, we invoke it on $c^{(0)}$, and then apply the functions given by the certification chain.

The distribution $\mathcal{D}_3$. This distribution is obtained from $\mathcal{D}_2$ by producing $\mathsf{crs}^{(0)}$ and π^* (where $c^* = (c_0^*, c_1^*, \pi^*)$) using the simulator of the NIZK proof system $\Pi^{(0)}$.

The distribution $\mathcal{D}_4$. This distribution is obtained from $\mathcal{D}_3$ by producing the challenge ciphertext $c^* = (c_0^*, c_1^*, \pi^*)$ with $c_0^* = \mathsf{Enc}_{pk_0}(m)$ (instead of $c_0^* = \mathsf{Enc}_{pk_0}(m')$ as in $\mathcal{D}_3$).

The distribution $\mathcal{D}_5$. This distribution is obtained from $\mathcal{D}_4$ by producing the challenge ciphertext $c^* = (c_0^*, c_1^*, \pi^*)$ with $c_1^* = \mathsf{Enc}_{pk_1}(m)$ (instead of $c_1^* = \mathsf{Enc}_{pk_1}(m')$ as in $\mathcal{D}_4$).

The distribution $\mathcal{D}_6$. This distribution is obtained from $\mathcal{D}_5$ by producing $\mathsf{crs}^{(0)}$ and π^* (where $c^* = (c_0^*, c_1^*, \pi^*)$) using the algorithms $\mathsf{CRSGen}^{(0)}$ and $\mathsf{P}^{(0)}$, respectively (and not by using the simulator of the NIZK proof system $\Pi^{(0)}$ as in $\mathcal{D}_5$).

The distribution $\mathcal{D}_7$. This is the distribution $\mathsf{Real}^{\mathsf{CPA}}_{\Pi',A,t,r,q}$.

Before proving that the above distributions are computationally indistinguishable, we first prove that S_2 fails to produce a certification chain with all but a negligible probability.

Lemma 4.2. *In distributions $\mathcal{D}_1, \ldots, \mathcal{D}_6$, whenever A_2 outputs a valid ciphertext, S_2 generates a certification chain with all but a negligible probability.*

Proof. Assume towards a contradictions that in one of $\mathcal{D}_1, \ldots, \mathcal{D}_6$ with a non-negligible probability it holds that A_2 outputs a valid ciphertext but S_2 fails to generate a certification chain. In particular, there exists an index $i \in \{1, \ldots, t\}$ for which with a non-negligible probability A_2 outputs a valid ciphertext of the form $c^{(i)} = \left(i, c_0^{(i)}, c_1^{(i)}, \pi^{(i)}\right)$, but S_2 fails to generate a certification chain. Recall that for generating a certification chain starting with $c^{(i)}$, the simulator S_2 attempts to invoke i knowledge extractors (until the first failure occurs) that we denote by $\mathsf{Ext}^{(i)}, \ldots, \mathsf{Ext}^{(1)}$. These knowledge extractors correspond to the malicious provers described in the description of S_2 for the argument systems $\Pi^{(i)}, \ldots, \Pi^{(1)}$, respectively. Then, there exists an index $j \in \{1, \ldots, i\}$ for which with a non-negligible probability S_2 is successful with $\mathsf{Ext}^{(i)}, \ldots, \mathsf{Ext}^{(j+1)}$ but fails with $\mathsf{Ext}^{(j)}$. The fact that S_2 is successful with $\mathsf{Ext}^{(j+1)}$ implies that it produces a valid ciphertext $c^{(j)} = \left(j, c_0^{(j)}, c_1^{(j)}, \pi^{(j)}\right)$. In particular, it holds that

$$\mathsf{V}^{(j)}\left(\left(pk_0, pk_1, c_0^{(j)}, c_1^{(j)}, \mathsf{crs}^{(j-1)}, \ldots, \mathsf{crs}^{(0)}\right), \pi^{(j)}, \mathsf{crs}^{(j)}\right) = 1 \ .$$

Now, the fact that with a non-negligible probability S_2 fails with $\mathsf{Ext}^{(j)}$ immediately translates to a malicious prover that contradicts the knowledge extraction property of the argument system $\Pi^{(j)}$. ■

We now prove that for every $i \in \{1, \ldots, 6\}$ the distributions $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$ are computationally indistinguishable.

Lemma 4.3. *The distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable.*

Proof. Whenever A_2 outputs an invalid ciphertext, or outputs a valid ciphertext and S_2 generates a certification chain, the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are identical. Indeed, in such a case the perfect decryption property guarantees that $\mathsf{Dec}'_{sk}\left(c^{(i)}\right) = f\left(m^{(0)}\right)$. Therefore, $\mathcal{D}_1$ and $\mathcal{D}_2$ differ only when when A_2 outputs a valid ciphertext but S_2 fails to generate a certification chain. Lemma 4.2 guarantees that this event occurs with only a negligible probability. ■

Lemma 4.4. *The distributions $\mathcal{D}_2$ and $\mathcal{D}_3$ are computationally indistinguishable.*

Proof. This follows from the zero-knowledge property of $\Pi^{(0)}$. Specifically, any efficient algorithm that distinguishes between $\mathcal{D}_2$ and $\mathcal{D}_3$ can be used (together with S) in a straightforward manner to contradict the zero-knowledge property of $\Pi^{(0)}$. ■

Lemma 4.5. *The distributions $\mathcal{D}_3$ and $\mathcal{D}_4$ are computationally indistinguishable.*

Proof. The simulation soundness of $\Pi^{(0)}$ guarantees that instead of computing $\mathsf{Dec}'_{sk}\left(c^{(0)}\right)$ we can verify that $\mathsf{V}^{(0)}\left(\left(pk_0, pk_1, c_0^{(0)}, c_1^{(0)}\right), \pi^{(0)}, \mathsf{crs}^{(0)}\right) = 1$, and then compute $\mathsf{Dec}_{sk_1}\left(c_1^{(0)}\right)$. The resulting distribution will be identical with all but a negligible probability. This implies that we do not need the key sk_0, and this immediately translates to a distinguisher between $(pk_0, \mathsf{Enc}_{pk_0}(m))$ and $(pk_0, \mathsf{Enc}_{pk_0}(m'))$, where m and m' are sampled independently from $\mathcal{M}$. That is, the simulation soundness of $\Pi^{(0)}$ and the semantic security of the underlying encryption scheme guarantee that $\mathcal{D}_3$ and $\mathcal{D}_4$ are computationally indistinguishable. ■

Lemma 4.6. *The distributions $\mathcal{D}_4$ and $\mathcal{D}_5$ are computationally indistinguishable.*

Proof. As in the proof of Lemma 4.5, the simulation soundness of $\Pi^{(0)}$ guarantees that instead of computing $\mathsf{Dec}'_{sk}(c^{(0)})$ we can verify that $\mathsf{V}^{(0)}\left(\left(pk_0, pk_1, c_0^{(0)}, c_1^{(0)}\right), \pi^{(0)}, \mathsf{crs}^{(0)}\right) = 1$, and then compute $\mathsf{Dec}_{sk_0}\left(c_0^{(0)}\right)$. The resulting distribution will be identical with all but a negligible probability. This implies that we do not need the key sk_1, and this immediately translates to a distinguisher between $(pk_1, \mathsf{Enc}_{pk_1}(m))$ and $(pk_1, \mathsf{Enc}_{pk_1}(m'))$, where m and m' are sampled independently from $\mathcal{M}$. That is, the simulation soundness of $\Pi^{(0)}$ and the semantic security of the underlying encryption scheme guarantee that $\mathcal{D}_4$ and $\mathcal{D}_5$ are computationally indistinguishable. ■

Lemma 4.7. *The distributions $\mathcal{D}_5$ and $\mathcal{D}_6$ are computationally indistinguishable.*

Proof. As in the proof of Lemma 4.4, this follows from the zero-knowledge property of $\Pi^{(0)}$. Specifically, any efficient algorithm that distinguishes between $\mathcal{D}_5$ and $\mathcal{D}_6$ can be used (together with S) in a straightforward manner to contradict the zero-knowledge property of $\Pi^{(0)}$. ■

Lemma 4.8. *The distributions $\mathcal{D}_6$ and $\mathcal{D}_7$ are computationally indistinguishable.*

Proof. In the distributions $\mathcal{D}_6$ and $\mathcal{D}_7$ the algorithm A_2 receives an encryption c^* of m, and outputs a ciphertext c. First, we note that in both $\mathcal{D}_6$ and $\mathcal{D}_7$, if $c = c^*$ then the output is $(\mathsf{state}_1, m, \mathsf{copy}_1)$, and if c is invalid then the output is $(\mathsf{state}_1, m, \perp)$. Therefore, we now focus on the case that $c \neq c^*$ and c is valid. In this case, in $\mathcal{D}_7$ the output is $(\mathsf{state}_1, m, \mathsf{Dec}'_{sk}(c))$, and we now show that with an overwhelming probability the same output is obtained also in $\mathcal{D}_6$.

In $\mathcal{D}_6$ Lemma 4.2 guarantees that whenever c is valid S_2 produces a certification chain with all but a negligible probability. There are now two cases to consider. In the first case, if $c^{(0)} = c^*$ and $i > 0$ then the output of $\mathcal{D}_6$ is $(\mathsf{state}_1, m, f(m))$, where $f = f^{(0)} \circ \cdots \circ f^{(i-1)}$. Since c^* is in fact an encryption of m, then the perfect decryption property guarantees that $\mathsf{Dec}'_{sk}(c) = f(m)$. In the second case, if $c^{(0)} \neq c^*$ then the output of $\mathcal{D}_6$ is $\left(\mathsf{state}_1, m, f\left(m^{(0)}\right)\right)$ where $m^{(0)} = \mathsf{Dec}'_{sk}\left(c^{(0)}\right)$. Again by the perfect decryption property, it holds that $\mathsf{Dec}'_{sk}(c) = f(m^{(0)})$. ■

This concludes the proof of Theorem 4.1. ■

4.4 Chosen-Ciphertext Security

We now show that the proof of security in Section 4.3 in fact extends to the setting of a-priori chosen-ciphertext attacks (CCA1). The difficulty in extending the proof is that now whereas the adversary A_1 is given oracle access to the decryption algorithm, the simulator S_1 is not given such access. Therefore, it is not immediately clear that S_1 can correctly simulate the decryption queries of A_1. We note that this issue seems to capture the main difference between the simulation-based and the indistinguishability-based approaches for defining non-malleability, as pointed out by Bellare and Sahai [BS99].

We resolve this issue using the approach of [DDN00, BS99]: S will not run A on the given public key pk, but instead will sample a new public key pk' together with a corresponding secret key sk', and run A on pk'. This way, S can use the secret key sk' for answering all of A_1's decryption queries. In addition, when A_2 outputs a ciphertext, S then uses sk' for "translating" this ciphertext from pk' to pk.

We now provide the modified description of S. Given an adversary $A = (A_1, A_2)$ we define the simulator $S = (S_1, S_2)$ as follows.

The algorithm S_1. The algorithm S_1 on input $(1^k, pk)$ first samples $(sk', pk') \leftarrow \mathsf{KeyGen}'(1^k)$. Then, it invokes A_1 on the input $(1^k, pk')$ while answering decryption queries using the secret key sk', to obtain a triplet $(\mathcal{M}, \mathsf{state}_1, \mathsf{state}_2)$. Finally, it outputs $(\mathcal{M}, \mathsf{state}_1, \mathsf{state}'_2)$ where $\mathsf{state}'_2 = (\mathcal{M}, pk, sk', pk', \mathsf{state}_2)$.

The algorithm S_2. The algorithm S_2 on input $(1^k, \mathsf{state}'_2)$ where $\mathsf{state}'_2 = (\mathcal{M}, pk, sk', pk', \mathsf{state}_2)$, first samples $m' \leftarrow \mathcal{M}$ and computes $c^* \leftarrow \mathsf{Enc}'_{pk'}(m')$. Then, it samples $r \leftarrow \{0,1\}^*$ and computes $c = \left(i, c_0^{(i)}, c_1^{(i)}, \pi^{(i)}\right) = A_2(1^k, c^*, \mathsf{state}_1; r)$. If c is invalid with respect to pk', that is, if $i \notin \{0, \ldots, t\}$ or

$$\mathsf{V}^{(i)}\left(\left(pk'_0, pk'_1, c_0^{(i)}, c_1^{(i)}, \mathsf{crs}'^{(i-1)}, \ldots, \mathsf{crs}'^{(0)}\right), \pi^{(i)}, \mathsf{crs}'^{(i)}\right) = 0$$

then S_2 outputs any ciphertext that is invalid with respect to pk (e.g., $(t+1, \bot, \bot, \bot)$). Otherwise, as in Section 4.3, S_2 utilizes the knowledge extractors guaranteed by the argument systems $\Pi^{(1)}, \ldots, \Pi^{(t)}$ to generate a "certification chain" for c of the form

$$\left(c^{(0)}, f^{(0)}, \ldots, c^{(i-1)}, f^{(i-1)}, c^{(i)}\right) \ .$$

If S_2 fails in generating such a chain then it again outputs any invalid ciphertext with respect to pk. Otherwise, S_2 computes its output as follows:

1. If $c^{(0)} = c^*$ and $i = 0$, then S_2 outputs copy_1.
2. If $c^{(0)} = c^*$ and $i > 0$, then S_2 outputs $f^{(0)} \circ \cdots \circ f^{(i-1)}$.
3. If $c^{(0)} \neq c^*$, then S_2 outputs the ciphertext $\widetilde{c}$ that is obtained by "translating" c from pk' to pk as follows. First, S_2 computes $\widetilde{m} = \mathsf{Dec}_{sk'}\left(c^{(0)}\right)$. Then, it computes $\widetilde{c}^{(0)} = \mathsf{Enc}_{pk}\left(\widetilde{m}\right)$, and $\widetilde{c}^{(j)} = \mathsf{HomEval}_{pk}\left(\widetilde{c}^{(j-1)}, f^{(j-1)}\right)$ for every $j \in \{1, \ldots, i\}$. The ciphertext $\widetilde{c}$ is then defined as $\widetilde{c}^{(i)}$.

Having described the modified simulator we now prove the following theorem stating the security of the scheme in the case $r(k) = q(k) = 1$ (noting once again that the more general case is a straightforward generalization).

Theorem 4.9. *For any constant $t \in \mathbb{N}$ and for any probabilistic polynomial-time adversary A the distributions $\{\mathsf{Real}^{\mathsf{CCA1}}_{\Pi',A,t,r,q}(k)\}_{k\in\mathbb{N}}$ and $\{\mathsf{Sim}^{\mathsf{CCA1}}_{\Pi',S,t,r,q}(k)\}_{k\in\mathbb{N}}$ are computationally indistinguishable, for $r(k) = q(k) = 1$.*

Proof. As in the proof of Theorem 4.1 we define a similar sequence of distributions $\mathcal{D}_1, \ldots, \mathcal{D}_7$ such that $\mathcal{D}_1 = \mathsf{Sim}^{\mathsf{CCA1}}_{\Pi',S,t,r,q}$ and $\mathcal{D}_7 = \mathsf{Real}^{\mathsf{CCA1}}_{\Pi',A,t,r,q}$, and prove that for every $i \in \{1, \ldots, 6\}$ the distributions $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$ are computationally indistinguishable. The main difference is that in this case we change the distribution of the public key pk' chosen by the simulator, and not of the given public key pk. The proofs are very similar, and therefore here we only point out the main differences.

The distribution $\mathcal{D}_1$. This is the distribution $\mathsf{Sim}^{\mathsf{CCA1}}_{\Pi',S,t,r,q}$.

The distribution $\mathcal{D}_2$. This distribution is obtained from $\mathcal{D}_1$ via the following modification. As in $\mathcal{D}_1$, if S_2 fails to obtain a certification chain, then output $(\mathsf{state}_1, m, \perp)$. Otherwise, the output is computed as follows:

1. If $c^{(0)} = c^*$ and $i = 0$ then output $(\mathsf{state}_1, m, \mathsf{copy}_1)$. This is identical to $\mathcal{D}_1$.
2. If $c^{(0)} = c^*$ and $i > 0$ then output $(\mathsf{state}_1, m, f(m))$, where $f = f^{(0)} \circ \cdots \circ f^{(i-1)}$. This is identical to $\mathcal{D}_1$.
3. If $c^{(0)} \neq c^*$ then compute $m^{(0)} = \mathsf{Dec}'_{sk'}\left(c^{(0)}\right)$. If $m^{(0)} \neq \perp$ output $\left(\mathsf{state}_1, m, f\left(m^{(0)}\right)\right)$, where $f = f^{(0)} \circ \cdots \circ f^{(i-1)}$, and otherwise output $(\mathsf{state}_1, m, \perp)$. That is, in this case instead of invoking the decryption algorithm Dec' on $c^{(i)}$, we invoke it on $c^{(0)}$, and then apply the functions given by the certification chain.

The distribution $\mathcal{D}_3$. This distribution is obtained from $\mathcal{D}_2$ by changing the distribution of pk' (that is chosen by S) and the challenge ciphertext: we produce $\mathsf{crs}'^{(0)}$ and π^* (where $c^* = (c^*_0, c^*_1, \pi^*)$) using the simulator of the NIZK proof system $\Pi^{(0)}$.

The distribution $\mathcal{D}_4$. This distribution is obtained from $\mathcal{D}_3$ by producing the challenge ciphertext $c^* = (c^*_0, c^*_1, \pi^*)$ with $c^*_0 = \mathsf{Enc}_{pk'_0}(m)$ (instead of $c^*_0 = \mathsf{Enc}_{pk'_0}(m')$ as in $\mathcal{D}_3$).

The distribution $\mathcal{D}_5$. This distribution is obtained from $\mathcal{D}_4$ by producing the challenge ciphertext $c^* = (c^*_0, c^*_1, \pi^*)$ with $c^*_1 = \mathsf{Enc}_{pk'_1}(m)$ (instead of $c^*_1 = \mathsf{Enc}_{pk'_1}(m')$ as in $\mathcal{D}_4$).

The distribution $\mathcal{D}_6$. This distribution is obtained from $\mathcal{D}_5$ by changing the distribution of pk' (that is chosen by S) and the challenge ciphertext: we produce $\mathsf{crs}'^{(0)}$ and π^* (where $c^* = (c^*_0, c^*_1, \pi^*)$) using the algorithms $\mathsf{CRSGen}^{(0)}$ and $\mathsf{P}^{(0)}$, respectively (and not by using the simulator of the NIZK proof system $\Pi^{(0)}$ as in $\mathcal{D}_5$).

The distribution $\mathcal{D}_7$. This is the distribution $\mathsf{Real}^{\mathsf{CPA}}_{\Pi',A,t,r,q}$.

The remainder of the proof is essentially identical to the proof of Theorem 4.1. The only subtle point is that S_1 can simulate the decryption oracle to A_1 while knowing only one of sk'_0 and sk'_1. Specifically, given a ciphertext $\left(i, c^{(i)}_0, c^{(i)}_1, \pi^{(i)}\right)$, it outputs $\perp$ if $i \notin \{0, \ldots, t\}$ or

$$\mathsf{V}^{(i)}\left(\left(pk'_0, pk'_1, c^{(i)}_0, c^{(i)}_1, \mathsf{crs}'^{(i-1)}, \ldots, \mathsf{crs}'^{(0)}\right), \pi^{(i)}, \mathsf{crs}'^{(i)}\right) = 0 \ .$$

Otherwise, it computes $m_b = \mathsf{Dec}_{sk'_b}\left(c^{(i)}_b\right)$ for the value $b \in \{0,1\}$ for which its known the key sk'_b. The soundness of the proof system $\Pi^{(0)}$ and of the argument systems $\Pi^{(1)}, \ldots, \Pi^{(t)}$ guarantee that the simulation is correct with all but a negligible probability. ■

5 The Tree-Based Construction

In this section we present our second construction which is obtained by modifying our first construction to offer a different trade-off between the length of the public key and the length of the ciphertext. As in Section 4, the scheme is parameterized by an upper bound t on the number of repeated homomorphic operations that can be applied to a ciphertext produced by the encryption algorithm. Recall that in our first construction, the length of the ciphertext is essentially independent of t, and the public key consists of $t+1$ common reference strings. In our second construction

the number of common reference strings in the public key is only $\log t$, and a ciphertext now consists of $\log t$ ciphertexts of the underlying homomorphic scheme and $\log t$ succinct arguments. Such a trade-off may be preferable over the one offered by our first construction, for example, when using argument systems that are tailored to the $\mathcal{NP}$ languages under considerations and or when it is not possible to use the same common reference string for all argument systems (depending, of course, on the length of the longest common reference strings).

The main idea underlying this construction is that the arguments computed by the homomorphic evaluation algorithm form a tree structure instead of a path structure. Specifically, instead of using t argument systems, we use only $d = \log t$ argument systems where the i-th one is used for arguing the well-formedness of a ciphertext after 2^i repeated homomorphic operations.

In Section 5.1 we formally specify the building blocks of the scheme, and in Section 5.2 we provide a description of the scheme and discuss its proof of security against a-priori chosen-ciphertext attacks (CCA1), which is rather similar to that of our first construction.

5.1 The Building Blocks

Our construction relies on the following building blocks:

1. A homomorphic public-key encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{HomEval})$ with respect to an efficiently recognizable set of efficiently computable functions $\mathcal{F}$. We assume that the scheme has almost-all-key perfect decryption (see Definition 2.1). In addition, as discussed in Section 2.1, as we do not consider function privacy we assume without loss of generality that $\mathsf{HomEval}$ is deterministic.

   For any security parameter $k \in \mathbb{N}$ we denote by $\ell_{pk} = \ell_{pk}(k)$, $\ell_m = \ell_m(k)$, $\ell_r = \ell_r(k)$, and $\ell_c = \ell_c(k)$ the bit-lengths of the public key, plaintext, randomness of Enc, and ciphertext[7], respectively, for the scheme Π. In addition, we denote by $\mathsf{V}_{\mathcal{F}}$ the deterministic polynomial-time algorithm for testing membership in the set $\mathcal{F}$, and denote by $\ell_{\mathcal{F}} = \ell_{\mathcal{F}}(k)$ the bit-length of the description of each function $f \in \mathcal{F}$.

2. A non-interactive deterministic-verifier simulation-sound adaptive zero-knowledge proof system (see Section 2.3) $\Pi^{(0)} = \left(\mathsf{CRSGen}^{(0)}, \mathsf{P}^{(0)}, \mathsf{V}^{(0)}\right)$ for the $\mathcal{NP}$-language $L^{(0)} = \bigcup_{k\in\mathbb{N}} L^{(0)}(k)$ defined as follows.

$$L^{(0)}(k) = \left\{ \left(pk_0, pk_1, c_0^{(0)}, c_1^{(0)}\right) \in \{0,1\}^{2\ell_{pk}+2\ell_c} : \begin{array}{c} \exists (m, r_0, r_1) \in \{0,1\}^{\ell_m+2\ell_r} \text{ s.t.} \\ c_0^{(0)} = \mathsf{Enc}_{pk_0}(m; r_0) \\ \text{and } c_1^{(0)} = \mathsf{Enc}_{pk_1}(m; r_1) \end{array} \right\}$$

   For any security parameter $k \in \mathbb{N}$ we denote by $\ell_{\mathsf{crs}^{(0)}} = \ell_{\mathsf{crs}^{(0)}}(k)$ and $\ell_{\pi^{(0)}} = \ell_{\pi^{(0)}}(k)$ the bit-lengths of the common reference strings produced by $\mathsf{CRSGen}^{(0)}$ and of the proofs produced by $\mathsf{P}^{(0)}$, respectively.

3. For every $j \in \{1, \ldots, d\}$ (where $d = \lceil \log t \rceil$) a 1/7-succinct non-interactive deterministic-verifier extractable argument system (see Section 2.2) $\Pi^{(j)} = \left(\mathsf{CRSGen}^{(j)}, \mathsf{P}^{(j)}, \mathsf{V}^{(j)}\right)$ for the

[7] For simplicity we assume a fixed upper bound ℓ_c on the length of ciphertexts, but this is not essential to our construction. More generally, one can allow the length of ciphertexts to increase as a result of applying the homomorphic operation.

$\mathcal{NP}$-language $L^{(j)} = \bigcup_{k\in\mathbb{N}} L^{(j)}(k)$ defined as follows for $j = 1$

$$L^{(1)}(k) = \left\{ \begin{array}{l} \left(pk_0, pk_1, c_0^{(1)}, c_1^{(1)}, c_0^{(2)}, c_1^{(2)}\right) \in \{0,1\}^{2\ell_{pk}+4\ell_c} : \\ \quad \exists f \in \{0,1\}^{\ell_{\mathcal{F}}} \text{ s.t.} \\ \quad\quad \bullet \;\; \mathsf{V}_{\mathcal{F}}(f) = 1 \\ \quad\quad \bullet \;\; c_0^{(2)} = \mathsf{HomEval}_{pk_0}\left(c_0^{(1)}, f\right) \\ \quad\quad \bullet \;\; c_1^{(2)} = \mathsf{HomEval}_{pk_1}\left(c_1^{(1)}, f\right) \end{array} \right\}$$

and defined as follows for $j > 1$:

$$L^{(j)}(k) = \left\{ \begin{array}{l} \left(pk_0, pk_1, c_0^{(1)}, c_1^{(1)}, c_0^{(2^j)}, c_1^{(2^j)}, \overrightarrow{\mathsf{crs}}^{(j-1)}\right) \in \{0,1\}^{\ell_1^{(j)}} : \\ \exists \left(c_0^{(2^{j-1})}, c_1^{(2^{j-1})}, c_0^{(2^{j-1}+1)}, c_1^{(2^{j-1}+1)}, f, \pi_L^{(j-1)}, \pi_R^{(j-1)}\right) \in \{0,1\}^{\ell_2^{(j)}} \text{ s.t.} \\ \quad \bullet \;\; \mathsf{V}_{\mathcal{F}}(f) = 1 \\ \quad \bullet \;\; c_0^{(2^{j-1}+1)} = \mathsf{HomEval}_{pk_0}\left(c_0^{(2^{j-1})}, f\right) \\ \quad \bullet \;\; c_1^{(2^{j-1}+1)} = \mathsf{HomEval}_{pk_1}\left(c_1^{(2^{j-1})}, f\right) \\ \quad \bullet \;\; \mathsf{V}^{(j-1)}\left(\left(pk_0, pk_1, c_0^{(1)}, c_1^{(1)}, c_0^{(2^{j-1})}, c_1^{(2^{j-1})}, \overrightarrow{\mathsf{crs}}^{(j-2)}\right), \pi_L^{(j-1)}, \mathsf{crs}^{(j-1)}\right) = 1 \\ \quad \bullet \;\; \mathsf{V}^{(j-1)}\left(\left(pk_0, pk_1, c_0^{(2^{j-1}+1)}, c_1^{(2^{j-1}+1)}, c_0^{(2^j)}, c_1^{(2^j)}, \overrightarrow{\mathsf{crs}}^{(j-2)}\right), \pi_R^{(j-1)}, \mathsf{crs}^{(j-1)}\right) = 1 \end{array} \right\}$$

where $\ell_1^{(j)} = 2\ell_{pk}+4\ell_c+\sum_{t=1}^{d-1}\ell_{\mathsf{crs}^{(t)}}$, $\ell_2^{(j)} = 4\ell_c+\ell_{\mathcal{F}}+2\ell_{\pi^{(j-1)}}$, and for every $1 \le j \le d$ we define $\overrightarrow{\mathsf{crs}}^{(j)} = \left(\mathsf{crs}^{(j)}, \ldots, \mathsf{crs}^{(1)}\right)$. For any security parameter $k \in \mathbb{N}$ we denote by $\ell_{\mathsf{crs}^{(j)}} = \ell_{\mathsf{crs}^{(j)}}(k)$ and $\ell_{\pi^{(j)}} = \ell_{\pi^{(j)}}(k)$ the bit-lengths of the common reference strings produced by $\mathsf{CRSGen}^{(j)}$ and of the arguments produced by $\mathsf{P}^{(j)}$, respectively.

We note that for $j = 1$ we in fact do not need an argument system, as we can use the witness $f \in \mathcal{F}$ as the arguments. Without loss of generality we assume that $\ell_{\pi^{(1)}} \ge \max\{\ell_c, \ell_{\mathcal{F}}\}$ (as otherwise arguments can always be padded). Thus, the assumption that the argument systems $\Pi^{(2)}, \ldots, \Pi^{(d)}$ are 1/7-succinct implies that the length of their arguments is upper bounded by that of $\Pi^{(1)}$ (and therefore independent of t).

5.2 The Scheme

The scheme $\Pi' = (\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{HomEval}')$ is parameterized by an upper bound t on the number of repeated homomorphic operations, and we let $d = \lceil \log t \rceil$. The key-generation and encryption algorithm are essentially identical to those described in Section 4.2:

- **Key generation:** On input 1^k sample two pairs of keys $(sk_0, pk_0) \leftarrow \mathsf{KeyGen}(1^k)$ and $(sk_1, pk_1) \leftarrow \mathsf{KeyGen}(1^k)$. Then, for every $j \in \{0, \ldots, d\}$ sample $\mathsf{crs}^{(j)} \leftarrow \mathsf{CRSGen}^{(j)}(1^k)$. Output the secret key $sk = (sk_0, sk_1)$ and the public key $pk = \left(pk_0, pk_1, \mathsf{crs}^{(0)}, \ldots, \mathsf{crs}^{(d)}\right)$.

- **Encryption:** On input a public key pk and a plaintext m, sample $r_0, r_1 \in \{0,1\}^*$ uniformly at random, and output the ciphertext $c^{(0)} = \left(c_0^{(0)}, c_1^{(0)}, \pi^{(0)}\right)$, where

$$\begin{aligned} c_0^{(0)} &= \mathsf{Enc}_{pk_0}(m; r_0) \;\; , \\ c_1^{(0)} &= \mathsf{Enc}_{pk_1}(m; r_1) \;\; , \\ \pi^{(0)} &\leftarrow \mathsf{P}^{(0)}\left(\left(pk_0, pk_1, c_0^{(0)}, c_1^{(0)}\right), (m, r_0, r_1), \mathsf{crs}^{(0)}\right) \;\; . \end{aligned}$$

- **Homomorphic evaluation:** The homomorphic evaluation algorithm follows the same approach used in Section 4.2, but computes the arguments of well-formedness in the form of a *sparse* binary tree. The leaves of the tree correspond to a chain of ciphertexts $\left(c^{(1)},\ldots,c^{(i)}\right)$ that are generated from one another using the homomorphic evaluation algorithm. Each internal node at level $j \in \{1,\ldots,d\}$ (where the leaves are considered to be at level 0) is a succinct argument for membership in the language $L^{(j)}$. We first describe how to generate the leaves and the internal nodes, and then describe the content of a ciphertext (i.e., which nodes of the tree should be contained in a ciphertext).

  - **The leaves:** The leftmost leaf in the tree $c^{(1)} = \left(c_0^{(1)}, c_1^{(1)}\right)$ is generated from a ciphertext $c^{(0)} = \left(c_0^{(0)}, c_1^{(0)}, \pi^{(0)}\right)$ that is produced by the encryption algorithm and a function $f^{(0)} \in \mathcal{F}$. It is defined as

    $$\begin{aligned} c_0^{(1)} &= \mathsf{HomEval}_{pk_0}\left(c_0^{(0)}, f^{(0)}\right) \ , \\ c_1^{(1)} &= \mathsf{HomEval}_{pk_1}\left(c_1^{(0)}, f^{(0)}\right) \ . \end{aligned}$$

    From this point on both the ciphertext $c^{(0)}$, the function $f^{(0)}$, and the leaf $c^{(1)}$ are kept part of all future ciphertexts.
    For every $i \in \{1,\ldots,t-1\}$ the leaf $c^{(i+1)} = \left(c_0^{(i+1)}, c_1^{(i+1)}\right)$ is generated from the previous leaf $c^{(i)} = \left(c_0^{(i)}, c_1^{(i)}\right)$ and a function $f^{(i)} \in \mathcal{F}$ by computing

    $$\begin{aligned} c_0^{(i+1)} &= \mathsf{HomEval}_{pk_0}\left(c_0^{(i)}, f^{(i)}\right) \ , \\ c_1^{(i+1)} &= \mathsf{HomEval}_{pk_1}\left(c_1^{(i)}, f^{(i)}\right) \ . \end{aligned}$$

  - **The internal nodes:** Each internal node v at level $j \in \{1,\ldots,d\}$ is an argument for membership in the language $L^{(j)}$. For $j = 1$, the two children of x are leaves $c^{(i)} = \left(c_0^{(i)}, c_1^{(i)}\right)$ and $c^{(i+1)} = \left(c_0^{(i+1)}, c_1^{(i+1)}\right)$, and in this case v is an argument that $c^{(i+1)}$ is obtained from $c^{(i)}$ using the homomorphic operation with some function $f^{(i)} \in \mathcal{F}$. This is computed as:

    $$\pi \leftarrow \mathsf{P}^{(1)}\left(\left(pk_0, pk_1, c_0^{(i)}, c_1^{(i)}, c_0^{(i+1)}, c_1^{(i+1)}, \right), f^{(i)}, \mathsf{crs}^{(1)}\right) \ .$$

    For every $j \in \{2,\ldots,d\}$, denote by v_L and v_R the two children of v. These are arguments for membership in $L^{(j-1)}$. Denote by $c^{(1)} = \left(c_0^{(1)}, c_1^{(1)}\right)$ and $c^{(2^{j-1})} = \left(c_0^{(2^{j-1})}, c_1^{(2^{j-1})}\right)$ the leftmost and rightmost leaves in the subtree of v_L, respectively. Similarly, denote by $c^{(2^{j-1}+1)} = \left(c_0^{(2^{j-1}+1)}, c_1^{(2^{j-1}+1)}\right)$ and $c^{(2^j)} = \left(c_0^{(2^j)}, c_1^{(2^j)}\right)$ the leftmost and rightmost leaves in the subtree of v_R, respectively. Then, the node v is an argument that v_L is a valid argument for $\left(c^{(1)},\ldots,c^{(2^{j-1})}\right)$, v_R is a valid argument for $\left(c^{(2^{j-1}+1)},\ldots,c^{(2^j)}\right)$, and that $c^{(2^{j-1}+1)}$ is obtained from $c^{(2^{j-1})}$ using the homomorphic operation with some function $f^{(2^{j-1})} \in \mathcal{F}$. This is computed as $\pi \leftarrow \mathsf{P}^{(j)}\left(x, w, \mathsf{crs}^{(j)}\right)$, where:

    $$\begin{aligned} x &= \left(pk_0, pk_1, c_0^{(1)}, c_1^{(1)}, c_0^{(2^j)}, c_1^{(2^j)}, \overrightarrow{\mathsf{crs}}^{(j-1)}\right) \\ w &= \left(c_0^{(2^{j-1})}, c_1^{(2^{j-1})}, c_0^{(2^{j-1}+1)}, c_1^{(2^{j-1}+1)}, f^{(2^{j-1})}, \pi_L^{(j-1)}, \pi_R^{(j-1)}\right) \ . \end{aligned}$$

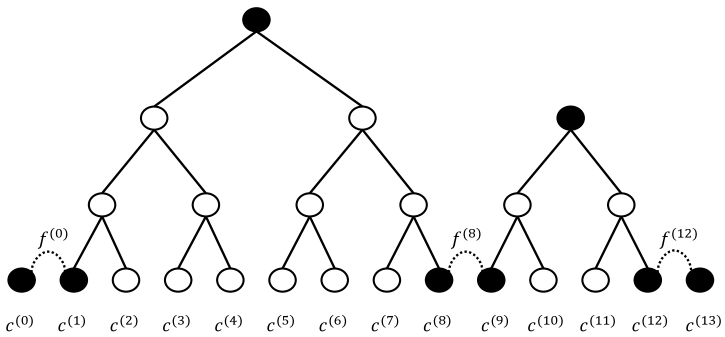

Figure 2: The above illustration shows the structure of a ciphertext after 13 repeated homomorphic operations. The ciphertext $c^{(0)}$ is an output of the encryption algorithm, and for every $i \in \{1, \ldots, 13\}$ the ciphertext $c^{(i)}$ is obtained from $c^{(i-1)}$ by applying the homomorphic evaluation algorithm using a function $f^{(i)} \in \mathcal{F}$. The internal nodes on levels 1, 2, and 3 contain succinct arguments for membership in the languages $L^{(1)}$, $L^{(2)}$, and $L^{(3)}$, respectively. The ciphertext of the new scheme consists of the black nodes and the functions $f^{(0)}$, $f^{(8)}$, and $f^{(12)}$.

  - **The ciphertext:** The ciphertext always includes the initial ciphertext $c^{(0)}$ that was produced by the encryption algorithm, the first leaf $c^{(1)}$, and the function $f^{(0)} \in \mathcal{F}$ that was used for generating $c^{(1)}$ from $c^{(0)}$. Then, every time we compute the value of two adjacent internal nodes v_L and v_R at some level $j-1$ that belong to the same subtree, we compute the value of their parent v at level j, as described above. As a result, we do not longer keep any information from the subtree of v, except for its leftmost and rightmost leaves. In addition, for every two adjacent subtrees we include the function that transforms the rightmost leaf of the first subtree to the leftmost leaf of the second subtree. Note that such subtrees must be of different depths, as otherwise they are merged. Thus, at any point in time a ciphertext may contain at most $2d+1$ ciphertexts of the underlying scheme, d short arguments, and d descriptions of functions from $\mathcal{F}$ (connecting subtrees). See Figure 2 for an illustration of the structure of a ciphertext.

- **Decryption:** On input the secret key sk and a ciphertext of the above form, verify the validity of the non-interactive zero-knowledge proof contained in $c^{(0)}$, verify that $c^{(1)}$ is obtained from $c^{(0)}$ using the function $f^{(0)} \in \mathcal{F}$, verify that the given tree has the right structure (with functions from $\mathcal{F}$ connecting subtrees), and verify the validity of all the arguments in the non-empty internal nodes of the tree. If any of these verifications fail, then output $\perp$. Otherwise, compute $m_0 = \mathsf{Dec}_{sk_0}\left(c_0^{(i)}\right)$ and $m_1 = \mathsf{Dec}_{sk_1}\left(c_1^{(i)}\right)$, where $c^{(i)} = \left(c_0^{(i)}, c_1^{(i)}\right)$ is the rightmost leaf. If $m_0 \neq m_1$ then output $\perp$, and otherwise output m_0.

Chosen-ciphertext security. As this scheme is obtained from the one in Section 4 by only changing the structure of the arguments that generate the ciphertext, the proof of security is rather similar to that in Sections 4.3 and 4.4. The only difference is in the way S_2 produces the "certification chain" for the ciphertext that the adversary outputs: instead of using the path structure of the ciphertext, the simulator now uses the tree structure of the ciphertext and applies the knowledge extractors accordingly. The remainder of the proof is exactly the same.

6 Extensions and Open Problems

We conclude the paper with a discussion of several extensions of our work and open problems.

The number of repeated homomorphic operations. Our schemes allow any pre-specified constant bound $t \in \mathbb{N}$ on the number of repeated homomorphic operations. It would be interesting to allow this bound to be a function $t(k)$ of the security parameter. As discussed in Section 2.2, the bottleneck is the super-linear length of the common-reference string in Groth's and Lipmaa's argument systems [Gro10, Lip11]. Any improvement to these argument systems with a common-reference string of linear length will directly allow any logarithmic number of repeated homomorphic operations in the path-based scheme, and any polynomial number of such operations in the tree-based scheme.

Function privacy and unlinkability. For some applications a homomorphic encryption scheme may be required to ensure *function privacy* [GHV10] or even *unlinkability* [PR08]. Function privacy asks that the homomorphic evaluation algorithm does not reveal (in a semantic security fashion) which operation it applies, and unlinkability asks that the output of the homomorphic evaluation algorithm is computationally indistinguishable from the output of the encryption algorithm. For example, the voting application discussed in the introduction requires function privacy to ensure that individual votes remain private. Our approach in this paper focuses on preventing a blow-up in the length of ciphertexts, and incorporating function privacy and unlinkability into our framework is an interesting direction for future work. We note that since Groth's argument system [Gro10] is also zero-knowledge it is quite plausible to show that ciphertexts in our schemes reveal nothing more than the number of repeated homomorphic operations.

A-posteriori chosen-ciphertext security (CCA2). As discussed in Section 1.3, Prabhakaran and Rosulek [PR08] considered the rather orthogonal problem of providing a homomorphic encryption scheme that is secure against a meaningful variant of a-posteriori chosen-ciphertext attacks (CCA2). In light of the fact that our schemes already offer targeted malleability against a-priori chosen-ciphertext attacks (CCA1), it would be interesting to extend our approach to the setting considered by Prabhakaran and Rosulek.

References

[ADR02] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *Advances in Cryptology – EUROCRYPT '02*, pages 83–107, 2002.

[AIK10] B. Applebaum, Y. Ishai, and E. Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming*, pages 152–163, 2010.

[BFM88] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 103–112, 1988.

[BP04a] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *Advances in Cryptology – CRYPTO '04*, pages 273–289, 2004.

[BP04b] M. Bellare and A. Palacio. Towards plaintext-aware public-key encryption without random oracles. In *Advances in Cryptology – ASIACRYPT '04*, pages 48–62, 2004.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BS99] M. Bellare and A. Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In *Advances in Cryptology – CRYPTO '99*, pages 519–536, 1999. The full version is available as Cryptology ePrint Archive, Report 2006/228.

[BSM+91] M. Blum, A. D. Santis, S. Micali, and G. Persiano. Noninteractive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991.

[CGS97] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology – EUROCRYPT '97*, pages 103–118, 1997.

[CKN03] R. Canetti, H. Krawczyk, and J. B. Nielsen. Relaxing chosen-ciphertext security. In *Advances in Cryptology – CRYPTO '03*, pages 565–582, 2003.

[CKV10] K.-M. Chung, Y. Kalai, and S. Vadhan. Improved delegation of computation using fully homomorphic encryption. In *Advances in Cryptology – CRYPTO '10*, pages 483–501, 2010.

[CS98] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology - CRYPTO '98*, pages 13–25, 1998.

[CT10] A. Chiesa and E. Tromer. Proof-carrying data and hearsay arguments from signature cards. In *Proceedings of the 1st Symposium on Innovations in Computer Science*, pages 310–331, 2010.

[Dam91] I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *Advances in Cryptology – CRYPTO '91*, pages 445–456, 1991.

[DDN00] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[DNR04] C. Dwork, M. Naor, and O. Reingold. Immunizing encryption schemes from decryption errors. In *Advances in Cryptology – EUROCRYPT '04*, pages 342–360, 2004.

[FLS90] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs based on a single random string. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 308–317, 1990.

[Gen09] C. Gentry. A fully homomorphic encryption scheme. PhD Thesis, Stanford University, 2009. Available at http://crypto.stanford.edu/craig.

[GGP10] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology – CRYPTO '10*, pages 465–482, 2010.

[GHV10] C. Gentry, S. Halevi, and V. Vaikuntanathan. i-hop homomorphic encryption and rerandomizable Yao circuits. In *Advances in Cryptology – CRYPTO '10*, pages 155–172, 2010.

[GKR08] S. Goldwasser, Y. T. Kalai, and G. Rothblum. Delegating computation: Interactive proofs for muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 113–122, 2008.

[Gro04] J. Groth. Rerandomizable and replayable adaptive chosen ciphertext attack secure cryptosystems. In *Proceedings of the 1st Theory of Cryptography Conference*, pages 152–170, 2004.

[Gro10] J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Advances in Cryptology – ASIACRYPT '10*, pages 321–340, 2010.

[GW11] C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, pages 99–108, 2011.

[Lin06] Y. Lindell. A simpler construction of CCA2-secure public-key encryption under general assumptions. *Journal of Cryptology*, 19(3):359–377, 2006.

[Lip11] H. Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. Cryptology ePrint Archive, Report 2011/009, 2011.

[Mic00] S. Micali. Computationally sound proofs. *SIAM Journal of Computing*, 30(4):1253–1298, 2000. An extended abstract appeared in *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, 1994.

[Nao03] M. Naor. On cryptographic assumptions and challenges. In *Advances in Cryptology – CRYPTO '03*, pages 96–109, 2003.

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, 1990.

[PR07] M. Prabhakaran and M. Rosulek. Rerandomizable RCCA encryption. In *Advances in Cryptology – CRYPTO '07*, pages 517–534, 2007.

[PR08] M. Prabhakaran and M. Rosulek. Homomorphic encryption with CCA security. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, pages 667–678, 2008.

[PSV07] R. Pass, A. Shelat, and V. Vaikuntanathan. Relations among notions of non-malleability for encryption. In *Advances in Cryptology - ASIACRYPT '07*, pages 519–535, 2007.

[RAD78] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 1978.

[Sah99] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 543–553, 1999.

[SV10] N. P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography – PKC '10*, pages 420–443, 2010.

[Val08] P. Valiant. Incrementally verifiable computation – or – proofs of knowledge imply time/space efficiency. In *Proceedings of the 5th Theory of Cryptography Conference*, pages 1–18, 2008.

[vDGH+10] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology – EUROCRYPT '10*, pages 24–43, 2010.