Generalizing Homomorphic MACs for Arithmetic Circuits

Dario Catalano[1], Dario Fiore[2], Rosario Gennaro[3], and Luca Nizzardo[4]*

[1] Università di Catania, Italy. catalano@dmi.unict.it
[2] IMDEA Software Institute, Spain. dario.fiore@imdea.org
[3] City College of New York, USA. rosario@cs.ccny.cuny.edu
[4] Università degli Studi di Milano-Bicocca, Italy. l.nizzardo@campus.unimib.it

Abstract. Homomorphic MACs, introduced by Gennaro and Wichs in 2013, allow anyone to validate computations on authenticated data without knowledge of the secret key. Moreover, the secret-key owner can verify the validity of the computation without needing to know the original (authenticated) inputs. Beyond security, homomorphic MACs are required to produce short tags (succinctness) and to support composability (i.e., outputs of authenticated computations should be re-usable as inputs for new computations).

At Eurocrypt 2013, Catalano and Fiore proposed two realizations of homomorphic MACs that support a restricted class of computations (arithmetic circuits of polynomial degree), are practically efficient, but fail to achieve both succinctness and composability at the same time.

In this paper, we generalize the work of Catalano and Fiore in several ways. First, we abstract away their results using the notion of encodings with limited malleability, thus yielding new schemes based on different algebraic settings. Next, we generalize their constructions to work with graded encodings, and more abstractly with k-linear groups. The main advantage of this latter approach is that it allows for homomorphic MACs which are (somewhat) composable while retaining succinctness. Interestingly, our construction uses graded encodings in a generic way. Thus, all its limitations (limited composability and non-constant size of the tags) solely depend on the fact that currently known multilinear maps share similar constraints. This means, for instance, that our scheme would support arbitrary circuits (polynomial depth) if we had compact multilinear maps with an exponential number of levels.

1 Introduction

Following the recent development of cloud computing, it is becoming popular for users to delegate the storage of their data to remote service providers. On one hand, this paradigm presents several benefits. For instance, users can access the data from different devices and different places. Moreover, even devices with limited storage capacity (e.g., smartphones) can have access to large amounts

[*] Work done while visiting CUNY.

of data. On the other hand, outsourcing data to remote (possibly untrusted) providers exposes the users to severe risks of privacy and integrity. While the community has devoted a lot of effort to finding ways to solve the privacy issue (notably the ground-breaking work on fully-homomorphic encryption [29]), the problem of integrity has received less attention. In particular, in this work we consider the following problem. Imagine that Alice wants to outsource a large amount of data to the cloud so that she can later (reliably) delegate the cloud to perform computation on such data. By "reliably" here we mean that the cloud should perform the computation and also be able to convince Alice that the computation was carried out correctly. What makes this task non trivial is that Alice does not keep a local copy of her data (i.e., the input of the computation) and that the communication complexity of the protocol should not depend on the size of the input. The latter restriction, for instance, rules out trivial solutions in which Alice can send signed data to the cloud and then ask the same (signed) data back in order to rerun the computation locally.

To solve this problem Gennaro and Wichs [28] put forward the notion of *homomorphic message authenticators* (homomorphic MACs, for short). Informally, a homomorphic MAC allows anyone, *without* knowledge of the secret key, to validate computations on authenticated data, and allows the secret-key owner to verify the results of these computations *without* knowing the original authenticated inputs. Slightly more in detail, a homomorphic MAC scheme enables a user to use his secret key to generate a tag σ that authenticates a message m. Later, given tags $\sigma_1, \ldots, \sigma_n$ for messages $m_1, \ldots, m_n$, *anyone* can run a program $\mathcal{P}$ over $\sigma_1, \ldots, \sigma_n$ to generate a short tag that authenticates (the output of) $\mathcal{P}(m_1, \ldots, m_n)$. To properly formalize the idea of authenticating a program's output, Gennaro and Wichs introduced the notion of *labeled* data and programs. The label of some message m is simply some string τ, which is used as auxiliary information to authenticate m. Intuitively, one can think of labels as names (or indexing) of the data. For instance, if a company outsources a database with information on its employees, the label "$(salary, i)$" might be used to indicate the salary value in the record corresponding to employee i. A *labeled program* $\mathcal{P}$ generalizes labeling to computations as follows. $\mathcal{P}$ is defined by a circuit f and a set of labels $(\tau_1 \ldots, \tau_n)$, one for each of the circuit's input wires. Intuitively, labeled programs provide a way to specify on which inputs the circuit has to be evaluated, without having to specify the exact values for such inputs. Basically, input labels can be seen as variable names in programming languages. In this sense, given a labeled program $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ and a set of tags $\sigma_1, \ldots, \sigma_n$—each authenticating m_i under label τ_i—anybody can run the (homomorphic) evaluation algorithm $\sigma \leftarrow \mathsf{Eval}(\mathcal{P}, \sigma_1, \ldots, \sigma_n)$ to obtain a tag σ that authenticates $m = \mathcal{P}(m_1, \ldots, m_n)$ as the output of $\mathcal{P}$ run on inputs labeled by $\tau_1, \ldots, \tau_n$ respectively.

Homomorphic MACs are required to satisfy three main properties. (1) They must be *secure*, i.e., an adversary that can (adaptively) see the tags corresponding to polynomially many messages of his own choice, should not be able to produce valid tags for messages that are not produced as the output of $\mathcal{P}$. (2) A homomorphic MAC should be *succinct*, in the sense that the authenticity of $\mathcal{P}$'s output should be certifiable using much less communication than what required to send the original inputs. (3) Finally, a homomorphic MAC should be *composable*, in the sense that tags authenticating previous computations should be usable as inputs to further authenticate new computations, i.e., computations executed on the results of other computations.

In terms of realizations, Gennaro and Wichs [28] proposed a *fully* homomorphic MAC scheme that achieves all the above three properties for arbitrary programs. On the negative side, their construction is unfortunately rather inefficient as it relies on the full power of fully homomorphic encryption. Moreover, it guarantees security only with respect to adversaries that are allowed to ask a constant number of verification queries. In recent work [16], Catalano and Fiore proposed a realization of homomorphic MACs that, while less general than [28], is more interesting from a practical point of view: it is more efficient, it guarantees security for an unbounded number of verification queries, and it can be based on minimal assumptions (OWFs). On the negative side, this efficiency gain comes at the cost of a somewhat reduced flexibility. More precisely, in [16] two solutions are proposed. The first one achieves full composability but guarantees succinctness only for circuits of low degree. The second construction, instead, achieves succinctness but does not guarantee fully-fledged composability[5].

Our Contribution. In this paper, we generalize the work of Catalano and Fiore, by proposing new constructions and extensions for their paradigm. In particular, our contribution is threefold.

First, we devise a general methodology to construct succinct homomorphic MACs using the notion of *encodings with limited malleability* first introduced in [8]. Very informally, these are encodings that are additively homomorphic and, at the same time, believed *not* to be multiplicative homomorphic. A bit more precisely, for the case of deterministic encodings, we require that given the econding of the ℓ powers of a random x, it must be computationally hard to come up with the encoding of $1/x$. We show that by encoding x as g^x, then the original scheme in [16] can be seen as an instance of our abstraction.

By replacing g^x in [16] with $\mathsf{Enc}(x)$, and presenting encodings based on different intractability assumptions, we then obtain new homomorphic MAC schemes, relying on such assumptions. In particular, we discuss encoding instantiations based on partially-homomorphic encryption schemes, such as Paillier [36], Boneh-Goh-Nissim [13] and Brakerski-Vaikuntanathan [15]. We remark that all such schemes, constitute examples of *randomized* encodings. In order for our proofs to go through, however, we need to be able to check when two encodings encode the same element. To accommodate this, our security assumption must

[5] This second scheme guarantees what the authors call *local composability*. In a nutshell, local composability allows to arbitrarily compose programs only when all the compositions are performed by the same entity.

be strengthened to assume that computing an encoding of $1/x$ remains hard *even if* these encryption schemes are *not* semantically secure[6].

The final resulting assumption (which we call ℓ-inversion resistance in the paper) is quite strong and somewhat non-standard: it is "non-constant" as it depends on the parameter ℓ, and also requires, in the security simulation, the hypothetical existence of a "distinguisher" that breaks the semantic security of the underlying encryption schemes. Yet assumptions of this type have been regularly used for many protocols in this area (e.g. [8, 26], cf. Footnote 6), and an intriguing open problem is to figure out how necessary they are.

For the same reason as in [16], though, the use of encodings for obtaining compact tags undermines the composability property. Our second contribution is a solution to this issue, which is obtained by further generalizing the idea of encodings with limited malleability. In particular, we build on so-called *graded encodings*, a notion recently proposed by Garg, Gentry and Halevi [24], that also provides an "approximate" realization of (leveled) multilinear groups. Basically, graded encodings are encodings that are additively homomorphic in the usual sense and multiplicatively homomorphic in a limited sense.[7] Our second construction uses graded encodings to obtain a homomorphic MAC scheme that achieves both composability and succinctness at the same time. In particular, if we instantiate our MAC by using the GGH graded encoding, we then obtain a scheme that allows for the following process: (1) one can generate constant-size tags, each authenticating the results of a computation $y_i = f_i(m_1, \ldots, m_n)$ for $i = 1$ to t, where each f_i is an arithmetic circuit of degree at most $D = \mathsf{poly}(\lambda)$; (2) one can compose the above computations f_i by using a "composition circuit" ϕ of degree at most k. Namely, one can finally authenticate $y = \phi(y_1, \ldots, y_t) = f^*(m_1, \ldots, m_n)$, where $f^* = \phi(f_1, \ldots, f_t)$. Here k is the degree of the multilinear groups. The size of the produced tags is linear in the degree of the composition circuit ϕ. Compared to the scheme of Catalano and Fiore, ours supports the same class of computations (i.e., arithmetic circuits of polynomial degree) but enjoys a higher degree of composability, and preserves succinctness as long as the composition circuit is low-degree.

Finally, we observe that our second scheme discussed above is *generically* built from multilinear maps. In particular, all its limitations (bounded circuits and size of the tags) are inherited from the current realizations of multilinear maps (e.g., GGH encodings): we support circuits of polynomial degree because the maps have polynomially-many levels, and our tags have size linear in the de-

[6] There is no contradiction here, as we are requiring the adversary to *compute* something ($\mathsf{Enc}(1/x)$) even if it is easy to decide if a given value t is an encoding of 0 or not. A similar situation arises in the SNARK protocols of [8, 26], when implemented with a randomized encoding – however in their case the assumptions made on the encoding are knowledge, non-falsifiable, ones (i.e. it is hard to compute $t = enc(x)$ with x satisfying certain constraints, without knowing x). Our assumption is falsifiable and conceptually simpler, though we do not know of any reduction from one to the other.

[7] Roughly speaking, the multiplicative homomorphism is limited in the sense that the result of a multiplication lies into a different encoding set.

gree of the composition circuit because the maps are not compact. This means that our scheme would support arbitrary circuits (polynomial depth) if we had compact multilinear maps with an exponential number of levels. Furthermore, our generic construction is proven secure against adversaries making an unbounded number of verification queries, in contrast to the fully-homomorphic MAC of Gennaro-Wichs, that can support only a constant number of verification queries. Therefore, although such powerful algebraic tools are not known, our result has the potential of yielding a fully-fledged homomorphic MAC.

Related Work. The problem of realizing homomorphic (mostly linear) authenticators either in the symmetric setting (MACs) or in the asymmetric one (signatures) has been the subject of several recent papers, starting with the seminal work of Johnson et al. [33]. The subject became popular more recently, due to an important application of homomorphic signatures to linear network coding. Efficient schemes of this primitive have been proposed both in the random oracle [10, 27, 12, 17] and in the standard model [1, 2, 18, 19, 23, 3]. Realizations for more complex functionalities (i.e., beyond linear) have been proposed only very recently [11, 28, 16, 4]. In addition to the works of Gennaro-Wichs [28] and Catalano-Fiore [28] that we already discussed in the previous section, it is worth mentioning two more works that are closely related to ours. Boneh and Freeman defined the notion of (fully) homomorphic signatures and showed a realization for bounded (constant) degree polynomials, from ideal lattices [11]. Compared to our work (and more in general to homomorphic MACs) this solution has the obvious advantage of allowing for public verifiability. On the negative side, however, it is not truly practical and the bound on the degree of the supported polynomials is more stringent than in our case (as they can support only polynomials of constant degree). Very recently, Backes, Fiore and Reischuk [4] put forward the notion of homomorphic MACs with efficient verification, which extends homomorphic MACs by requiring the verification algorithm to run more efficiently than the time necessary to compute the program $\mathcal{P}$ against which it verifies (precisely, they require amortized constant time). In [4], they propose a construction of this primitive based on the decision linear assumption and show applications to verifiable delegation of computation on outsourced data.

Other related work. Homomorphic signatures could be realized by using *Succinct Non-interactive Arguments of Knowledge* (SNARKs) [6]. Informally, a SNARK allows to construct a succinct argument that can be used to prove knowledge of the witness of a given any NP statement. SNARKs enjoy the nice property that the size of the proof is independent of the size of both statement and witness. A drawback of SNARKs is that they are not very efficient in practice and require either the random oracle model [35] or non-standard, non-falsifiable assumptions [30]. Moreover, the composability of homomorphic signatures obtained via SNARKs seems to be very limited [39, 7].

Homomorphic authenticators are also related to memory delegation [20] and verifiable computation [34, 35, 31, 25, 5, 37, 22]. We refer to [28] for a detailed discussion about similarities with these primitives.

2 Background and Definitions

In our work we use the notion of arithmetic circuits and related definitions. For lack of space, we refer the interested reader to [38] for a nice survey on this subject.

2.1 Homomorphic Message Authenticators

Labeled Programs. First, let us recall the notion of labeled programs introduced by Gennaro and Wichs in [28]. A labeled program $\mathcal{P}$ is defined by a tuple $(f, \tau_1, \ldots, \tau_n)$ where $f : \mathbb{F}^n \to \mathbb{F}$ is a circuit, and the binary strings $\tau_1, \ldots, \tau_n \in \{0,1\}^*$ are the *labels* of the input nodes of f. Given some labeled programs $\mathcal{P}_1, \ldots, \mathcal{P}_t$ and a function $g : \mathbb{F}^t \to \mathbb{F}$, the *composed program* $\mathcal{P}^* = g(\mathcal{P}_1, \ldots, \mathcal{P}_t)$ is the circuit which evaluates a circuit g on the outputs of $\mathcal{P}_1, \ldots, \mathcal{P}_t$ respectively. The labeled inputs of $\mathcal{P}^*$ are all distinct labeled inputs of $\mathcal{P}_1, \ldots, \mathcal{P}_t$, i.e., all inputs with the same label are put together in a single input of the new program. We denote with $\mathcal{I}_\tau = (g_{id}, \tau)$ the *identity program* with label τ where g_{id} is the canonical identity function and $\tau \in \{0,1\}^*$ is some input label. Notice that any program $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ can be expressed as the composition of n identity programs $\mathcal{P} = f(\mathcal{I}_{\tau_1}, \ldots, \mathcal{I}_{\tau_n})$.

Homomorphic Authenticator Scheme. A homomorphic message authenticator scheme HomMAC consists of the following four algorithms:

$\mathsf{KeyGen}(1^\lambda)$**:** on input the security parameter λ, the key generation algorithm outputs a secret key sk and a public evaluation key ek.

$\mathsf{Auth}(\mathsf{sk}, \tau, m)$**:** given the secret key sk, an input-label τ and a message $m \in \mathcal{M}$, it outputs a tag σ.

$\mathsf{Ver}(\mathsf{sk}, m, \mathcal{P}, \sigma)$**:** given the secret key sk, a message $m \in \mathcal{M}$, a program $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ and a tag σ, the verification algorithm outputs 0 (reject) or 1 (accept).

$\mathsf{Eval}(\mathsf{ek}, f, \boldsymbol{\sigma})$**:** given the evaluation key ek, a circuit $f : \mathcal{M}^n \to \mathcal{M}$ and a vector of tags $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$, the evaluation algorithm outputs a new tag σ.

Authentication Correctness. Intuitively, a homomorphic MAC satisfies this property if any tag σ generated by the algorithm $\mathsf{Auth}(\mathsf{sk}, \tau, m)$ authenticates with respect to the identity program $\mathcal{I}_\tau$. Formally, we require that for any message $m \in \mathcal{M}$, all keys $(\mathsf{sk}, \mathsf{ek}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$, any label $\tau \in \{0,1\}^*$, and any tag $\sigma \overset{\$}{\leftarrow} \mathsf{Auth}(\mathsf{sk}, \tau, m)$, it holds: $\Pr[\mathsf{Ver}(\mathsf{sk}, m, \mathcal{I}_\tau, \sigma) = 1] = 1$.

Evaluation Correctness. Informally, this property states that if the evaluation algorithm is given a vector of tags $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$ such that each σ_i authenticates some message m_i as the output of a labeled program $\mathcal{P}_i$, then the tag σ produced by Eval must authenticate $f(m_1, \ldots, m_n)$ as the output of the composed program $f(\mathcal{P}_1, \ldots, \mathcal{P}_n)$.

More formally, let us fix a pair of keys $(\mathsf{sk}, \mathsf{ek}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$, a function $g : \mathcal{M}^t \to \mathcal{M}$ and any set of message/program/tag triples $\{(m_i, \mathcal{P}_i, \sigma_i)\}_{i=1}^t$ such that $\mathsf{Ver}(\mathsf{sk}, m_i, \mathcal{P}_i, \sigma_i) = 1$. If $m^* = g(m_1, \ldots, m_t)$, $\mathcal{P}^* = g(\mathcal{P}_1, \ldots, \mathcal{P}_t)$, and $\sigma^* = \mathsf{Eval}(\mathsf{ek}, g, (\sigma_1, \ldots, \sigma_t))$, then it must hold: $\mathsf{Ver}(\mathsf{sk}, m^*, \mathcal{P}^*, \sigma^*) = 1$.

Succinctness. The size of a tag is bounded by some fixed polynomial in the security parameter, that is independent of the number of inputs taken by the evaluated circuit.

Security. Here we recall the security definition of homomorphic MACs proposed by Catalano and Fiore [16] (which slightly weakens the one of Gennaro and Wichs [28]). A homomorphic MAC scheme HomMAC is secure if for any PPT adversary $\mathcal{A}$ we have $\Pr[\mathsf{HomUF{-}CMA}_{\mathcal{A},\mathsf{HomMAC}}(\lambda) = 1] \leq \epsilon(\lambda)$ where $\epsilon(\lambda)$ is a negligible function, and $\mathsf{HomUF{-}CMA}_{\mathcal{A},\mathsf{HomMAC}}(\lambda)$ is the experiment below.

Setup The challenger generates $(\mathsf{sk}, \mathsf{ek}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$ and gives ek to $\mathcal{A}$. Also a list $T = \emptyset$ is initialized.

Authentication queries The adversary can adaptively ask for tags on label-message pairs of its choice. Given a query (τ, m), if $(\tau, m) \in T$ (i.e., the query was previously made), then the challenger replies with the same tag generated before. If T already contains a pair $(\tau, m') \in T$ with $m' \neq m$ (i.e., the label τ was already queried with a different message), then the challenger ignores the query. Otherwise, if $(\tau, m) \notin T$, the challenger computes $\sigma \xleftarrow{\$} \mathsf{Auth}(\mathsf{sk}, \tau, m)$, returns σ to $\mathcal{A}$ and updates the list $T = T \cup (\tau, m)$.

Verification queries The adversary is also given access to a verification oracle. $\mathcal{A}$ can submit a query $(m, \mathcal{P}, \sigma)$ and the challenger replies with the output of $\mathsf{Ver}(\mathsf{sk}, m, \mathcal{P}, \sigma)$.

Forgery When the adversary stops running, the experiment outputs 1 if one of the verification queries made by $\mathcal{A}$, say $(m^*, \mathcal{P}^*, \sigma^*)$, is a forgery.

The description of the experiment is thus concluded by defining what is a forgery. To this end, we first recall the notion of a *well-defined* program with respect to a list T. Informally, there are two ways for a program $\mathcal{P}^* = (f^*, \tau_1^*, \ldots, \tau_n^*)$ to be well-defined. Either all the τ_i^*'s are in T or, if there are some labels τ_i^* that are not in T, then the inputs associated with such labels are "ignored" by f^* when computing the output. In other words, inputs corresponding to labels not in T do not affect the behavior of f^* in any way.

More formally, a labeled program $\mathcal{P}^* = (f^*, \tau_1^*, \ldots, \tau_n^*)$ is *well defined with respect to* T if either one of the following two cases occurs:

1. there exists $i \in \{1, \ldots, n\}$ such that $(\tau_i^*, \cdot) \notin T$ (i.e., $\mathcal{A}$ never asked an authentication query with label τ_i^*), and $f^*(\{m_j\}_{(\tau_j, m_j) \in T} \cup \{\tilde{m}_j\}_{(\tau_j, \cdot) \notin T})$ outputs the same value for all possible choices of $\tilde{m}_j \in \mathcal{M}$;
2. T contains tuples $(\tau_1^*, m_1), \ldots, (\tau_n^*, m_n)$, for some messages $m_1, \ldots, m_n$.

In the experiment $\mathsf{HomUF{-}CMA}$, a tuple $(m^*, \mathcal{P}^*, \sigma^*)$ is a forgery if and only if $\mathsf{Ver}(\mathsf{sk}, m^*, \mathcal{P}^*, \sigma^*) = 1$ and one of the following conditions holds:

- *Type 1 Forgery:* $\mathcal{P}^*$ is not well-defined w.r.t. T.
- *Type 2 Forgery:* $\mathcal{P}^*$ is well defined w.r.t. T and $m^* \neq f^*(\{m_j\}_{(\tau_j, m_j) \in T})$, i.e., m^* is not the correct output of the labeled program $\mathcal{P}^*$ when executed on previously authenticated messages $(m_1, \ldots, m_n)$.

As already noted in [16], the experiment $\mathsf{HomUF-CMA}$ requires the challenger to recognize whether a program submitted by the adversary in a verification query is well-defined or not, but the latter check might not be doable in polynomial time, at least for certain classes of computations. Catalano and Fiore observe that this is not a problem for the class of arithmetic circuits of polynomial degree and over an exponentially large finite field. Here we give a simple proposition (for lack of space its proof appears in the full version) to show that testing whether a program is well-defined can be done even for arithmetic circuits of degree d, over a finite field of order p such that $d/p < 1/2$.[8]

Proposition 1. *Let $\lambda, n \in \mathbb{N}$ and let $\mathcal{F}$ be the class of arithmetic circuits $f : \mathbb{F}^n \to \mathbb{F}$ over a finite field $\mathbb{F}$ of order p and such that the degree of f is at most d, for $\frac{d}{p} < \frac{1}{2}$. Then, there exists a probabilistic algorithm that for any given $f \in \mathcal{F}$, decides if there exists $y \in \mathbb{F}$ such that $f(\boldsymbol{u}) = y, \forall \boldsymbol{u} \in \mathbb{F}^n$ (i.e., if f is constant) and is correct with probability at least $1 - 2^{-\lambda}$.*

Furthermore, for the same class of computations, we show that Type-1 forgeries essentially "collapse" into Type-2 forgeries. Namely, we show that any adversary winning in the experiment by producing a Type-1 forgery can be converted into another adversary that wins by producing a Type-2 forgery. This is formalized in the following proposition whose proof is deferred to the full version:

Proposition 2. *Let $\lambda \in \mathbb{N}$ be the security parameter, and let $\mathcal{F}$ be the class of arithmetic circuits $f : \mathbb{F}^n \to \mathbb{F}$ over a finite field $\mathbb{F}$ of order p and such that the degree of f is at most d, for $\frac{d}{p} < \frac{1}{2}$. Let $\mathcal{E}_b$ be the event that the adversary wins in experiment $\mathsf{HomUF-CMA}$ by producing a Type-b forgery (for $b = 1, 2$). Then, if for any adversary $\mathcal{B}$ we have that $\Pr[\mathsf{HomUF-CMA}_{\mathcal{B},\mathsf{HomMAC}}(\lambda) = 1 \wedge \mathcal{E}_2] \leq \epsilon$, then for any adversary $\mathcal{A}$ producing a Type-1 forgery it holds $\Pr[\mathsf{HomUF-CMA}_{\mathcal{A},\mathsf{HomMAC}}(\lambda) = 1 \wedge \mathcal{E}_1] \leq Q(\epsilon + 2^{-\lambda})$.*

3 Compact Homomorphic MACs Based on Encodings with Limited Malleability

In this section we describe a generalization of the scheme of Catalano and Fiore [16] which uses a more general encoding to compress the tags. First we define the encoding that we are going to use to compress the tags. We then show the compact scheme and prove its security. Finally we show that the scheme from [16] can be seen as an instance of this generalization, and we also present a different implementation based on partially-homomorphic encryption.

Limited Malleability Encoding. An encoding $\mathcal{E}$ consists of three algorithms $(\mathsf{EncGen}, \mathsf{Enc}, \mathsf{Test})$ defined as follows:

[8] For simplicity, we show this for $1/2$. The same argument can be extended to $d/p < 1/c$ for some constant c.

$\mathsf{EncGen}(1^\lambda)$**.** Given the security parameter λ, it outputs a pair of public/secret keys pk, dk, the message space $\mathbb{Z}_p$ where p is a prime of at least λ-bits, and an encoding space T. We denote with $+, \cdot$ the usual additions/multiplications over $\mathbb{Z}_p$, while T is assumed to be an abelian group under operation $\times$.

$\mathsf{Enc}(\mathsf{pk}, m)$**.** A possibly randomized algorithm which takes as input $m \in \mathbb{Z}_p$ and returns a value $t \in T$.

$\mathsf{Test}(\mathsf{dk}, m, t)$ A deterministic algorithm which on input $m \in \mathbb{Z}_p$ and $t \in T$ outputs 1 if $t \in \mathsf{Enc}(\mathsf{pk}, m)$, and 0 otherwise.

On the testing algorithm. We note that if the encoding algorithm is deterministic, the testing procedure can be easily carried out by re-encoding m and checking that is equal to t. Also, note that in this case there is no need of a secret key to test. This is the case for the discrete-log based encoding in [16]. For more general encodings where the encoding algorithm might be randomized, a secret decoding key dk might be needed to "decode" t and check that is equal to m.

Definition 1. *We say that an encoding is* additively homomorphic *if for all $m, m' \in \mathbb{Z}_p$, and for all $h \in \mathsf{Enc}(pk, m)$ and $h' \in \mathsf{Enc}(pk, m')$ we have that $h \times h' \in \mathsf{Enc}(pk, m + m')$.*

Limited Malleability. We now define our security assumption for the encodings $\mathcal{E}$ that we use in our scheme. Basically, we ask that given $t_i = \mathsf{Enc}(pk, x^i)$ for $i = 0, \ldots, \ell$ it must be hard to compute $t = \mathsf{Enc}(pk, 1/x)$, even in the presence of an oracle which decides if an element of T is an encoding of 0 or not. More formally, define the oracle $O(\mathsf{dk}, \tau)$ which answers ”yes” if $\tau \in \mathsf{Enc}(\mathsf{pk}, 0)$ and "no" otherwise. Then for any PPT (adversary) $\mathcal{A}$, consider the following experiment $\textbf{E-INV}_{\mathcal{A}}(\lambda, \ell)$:

1. $(\mathsf{pk}, \mathsf{dk}, p, T) \leftarrow \mathsf{EncGen}(1^\lambda)$;
2. $x \leftarrow \mathbb{Z}_p^*$;
3. $h_i \leftarrow \mathsf{Enc}(pk, x^i)$ for $i = 1, \ldots, \ell$;
4. $t \leftarrow \mathcal{A}^{O(\mathsf{dk}, \cdot)}(\mathsf{pk}, p, T, h_0, \ldots, h_\ell)$

We define $\mathcal{A}$'s advantage in winning the $\textbf{E-INV}_{\mathcal{A}}(\lambda, \ell)$ game as $\textbf{Adv}_{\mathcal{A}}^{\mathcal{E}-INV}(\lambda, \ell)$ $= \Pr[t \in \mathsf{Enc}(pk, 1/x)]$.

Definition 2. *We say that $\mathcal{E}$ is ℓ-inversion-resistant if for every PPT $\mathcal{A}$ we have that $\textbf{Adv}_{\mathcal{A}}^{\mathcal{E}-INV}(\lambda, \ell)$ is negligible in λ.*

The assumption states that *computing* $\mathsf{Enc}(\mathsf{pk}, 1/x)$ must be difficult *even if* we were able to efficiently *decide* if a string is the encoding of $1/x$ (because of homomorphic properties of $\mathcal{E}$ deciding if τ is an encoding of 0 is equivalent to deciding if τ is an encoding of an arbitrary element of $\mathbb{Z}_p$).

We remark that we do *not* require the existence of the oracle to implement the encoding and our scheme. It is just needed by the simulation (therefore making our assumption stronger than just computing $\mathsf{Enc}(\mathsf{pk}, 1/x)$).

The Scheme. We now describe a homomorphic MAC scheme that works for arithmetic circuits of polynomial degree D (but does not support composition).

The authentication tags produced by our construction have size which is independent of the size/depth of the circuit. The description of our scheme follows.

$\mathsf{KeyGen}(1^\lambda, D)$**.** Let λ be the security parameter, and D be a parameter of size $\mathsf{poly}(\lambda)$. The key generation works as follows. Run $\mathsf{EncGen}(1^\lambda)$ to generate $\mathsf{pk}, \mathsf{dk}, p, T$. We assume that our circuits work over $\mathbb{Z}_p$. Next, select $x \xleftarrow{\$} \mathbb{Z}_p$, a seed K of a pseudorandom function $F_K : \{0,1\}^* \to \mathbb{Z}_p$, and compute $h_i = \mathsf{Enc}(pk, x^i)$, for $i = 0$ to $D-1$. Output $\mathsf{sk} = (K, \mathsf{dk}, x), \mathsf{ek} = (h_0, \dots, h_{D-1})$.

$\mathsf{Auth}(\mathsf{sk}, \tau, m)$**.** To authenticate a message $m \in \mathbb{Z}_p$ with label $\tau \in \{0,1\}^\lambda$, compute $r_\tau = F_K(\tau)$, set $y_0 = m$, $y_1 = (r_\tau - m)/x \bmod p$, and output $\sigma = (y_0, y_1) \in \mathbb{Z}_p^2$. The authentication tags produced by Auth are interpreted as degree-1 polynomials $y(X) = y_0 + y_1 X$ over the ring $\mathbb{Z}_p[X]$.

$\mathsf{Eval}(\mathsf{ek}, f, \boldsymbol{\sigma})$**.** The first step of this algorithm is the same as the Eval algorithm of the homomorphic MACs construction based on OWFs proposed in [16]. The input is the evaluation key ek, an arithmetic circuit $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$, and a vector $\boldsymbol{\sigma}$ of tags $(\sigma_1, \dots, \sigma_n)$ where each σ_i is a polynomial $y^{(i)}(X) \in \mathbb{Z}_p[X]$. The first step is to compute the polynomial $y(X)$ obtained by (homomorphically) evaluating the circuit f over the polynomial ring $\mathbb{Z}_p[X]$, i.e., $y(X) \leftarrow f(y^{(1)}(X), \dots, y^{(n)}(X))$. Namely, additions and multiplications over $\mathbb{Z}_p$ are replaced by additions and multiplications of polynomials over $\mathbb{Z}_p[X]$. Let $y_0, \dots, y_d$ be the coefficients of the polynomial $y(X)$ (note that $d \leq D$). If $d = 1$ then the algorithm returns $\sigma = (y_0, y_1)$, otherwise it computes $\Lambda = \Pi_{i=0}^{d-1} h_i^{y_{i+1}}$ (where this product is computed in the group T defined by the encoding) and returns $\sigma = \Lambda$.

$\mathsf{Ver}(\mathsf{sk}, m, \mathcal{P}, \sigma)$**.** Let $\mathcal{P} = (f, \tau_1, \dots, \tau_n)$ be a labeled program, $m \in \mathbb{Z}_p$ and σ be a tag of either the form $(y_0, y_1) \in \mathbb{Z}_p^2$, or $\Lambda \in T$.

First, compute $\rho = f(r_{\tau_1}, \dots, r_{\tau_n})$ where $r_{\tau_i} \leftarrow F_K(\tau_i)$. Next, according to the form of σ perform the following checks:

1. If $\sigma = (y_0, y_1)$, then output 1 only if $\rho = y_0 + y_1 \cdot x \,\wedge\, y_0 = m$.
2. If $\sigma = \Lambda$, then let $t = \Lambda^x$ and output $\mathsf{Test}(\mathsf{dk}, \rho - m, t)$

It is not difficult to check that the scheme is correct by the construction of the polynomials $y(X)$ and the correctness of the encoding $\mathcal{E}$. The following theorem proves the security of our construction. For lack of space, the proof of Theorem 1 as well as a full proof of correctness appear in the full version.

Theorem 1. *If $\mathcal{E}$ is $(D-1)$-inversion resistant, and F is a pseudorandom function, then the scheme described in Section 3 is a secure homomorphic MAC.*

Possible Instantiations.

DISCRETE-LOG BASED ENCODING. We first show that the protocol of Catalano and Fiore [16] fits into the abstraction we just described. In their case the encoding algorithms are as follows:

- $\mathsf{EncGen}(1^\lambda)$ chooses a prime p of size at least λ, and a cyclic group T of order p, and a generator g for it. $\mathsf{pk} = \mathsf{dk} = (p, T, g)$. Note that the decoding key is not secret.

- $\mathsf{Enc}(\mathsf{pk}, m) = g^m$. Note that the encoding scheme is deterministic.
- $\mathsf{Test}(\mathsf{dk}, m, t)$ checks if $t = g^m$

The assumption that this encoding is ℓ-inversion resistant is equivalent to the ℓ-Diffie-Hellman inversion assumption [14]: given $g, g^x, g^{x^2}, \ldots, g^{x^\ell}$ it is hard to compute $g^{1/x}$. Note that the oracle which test if t is an encoding of 0 is trivially implemented by checking if $t = 1$, since the encoding is deterministic.

Partially Homomorphic Encryption Schemes. Any encryption scheme which is additively homomorphic over a prime field but is believed to be multiplicative homomorphic only up to a *constant* degree, constitutes a suitable candidate to be an ℓ-inversion resistant encoding. Let (KG, Enc, Dec) be such an encryption scheme, then:

- $\mathsf{EncGen}(1^\lambda)$ runs $KG(1^\lambda)$ to choose a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$. It sets pk to be the encoding public key and $\mathsf{dk} = \mathsf{sk}$ its secret decoding key.
- $\mathsf{Enc}(\mathsf{pk}, m) = Enc(\mathsf{pk}, m)$. Note that in this case the encoding scheme is randomized.
- $\mathsf{Test}(\mathsf{dk}, m, t)$ checks if $Dec(\mathsf{sk}, t) = m$

Examples of encryption schemes with such a partial homomorphic property include the “basic” version of the Brakerski and Vaikuntanathan FHE [15], Boneh, Goh and Nissim [13], and Paillier [36] schemes. [9] Note that to use these schemes in our protocol we need to require them to be ℓ-inversion resistant as defined earlier. This is a strong notion of security: we require them to be one-way in a strong sense (given encryptions of ℓ successive powers of x, it is impossible to come up with an encryption of $1/x$) *even in the presence of an oracle that breaks the semantic security of the scheme.*

4 A Compact Scheme with k-degree Composition

In this section we propose a homomorphic MAC based on multilinear groups. Compared to [16], the main advantage of this scheme is that it provides a way to both compress the tags and enable their (later) composition. Before describing the scheme, we first recall the definition of graded encoding and its abstraction of leveled multilinear maps.

Leveled Multilinear Maps and Graded Encodings. Informally speaking, a k-graded encoding system for a ring R includes a system of sets $\{S_i^{(\alpha)} \subset \{0,1\}^* : i \in [0,k], \alpha \in R\}$ such that for every fixed $i \in [0,k]$ the sets $\{S_i^{(\alpha)} : \alpha \in R\}$ are disjoint. The set $S_i^{(\alpha)}$ essentially contains the level-i encodings of $\alpha \in R$.

[9] Although Paillier and BGN schemes operate over the ring Z_N where N is the product of two large primes, note that the zero-divisors are only a negligible fraction of $\mathbb{Z}_N$. Moreover, it is hard to find such divisors if we assume that factoring N is hard. Therefore, it is not hard to see that with minor modifications the proof of Theorem 1 can be changed to accomodate this. More details appear in the full version.

As a first requirement, the system needs an algorithm to obtain an encoding $a_i \in S_i^{(\alpha)}$ of some ring element α (notice that such encoding can be randomized). Additionally, the encoding system is homomorphic in a graded sense. Namely, let us abuse notation and assume that every set $S_i^{(\alpha)}$ is a ring where $+, \cdot$ are the usual addition/multiplication operations. Then, for any $a_i \in S_i^{(\alpha)}$ and $b_i \in S_i^{(\beta)}$ we have $a_i + b_j \in S_i^{(\alpha+\beta)}$. Furthermore, for $a_i \in S_i^{(\alpha)}$ and $b_j \in S_j^{(\beta)}$ we have $a_i \cdot b_j \in S_{i+j}^{(\alpha \cdot \beta)}$, if $i + j \leq k$. Finally, the encoding system has an algorithm to test if a given a is an encoding of 0 at level i, i.e., $a \in S_i^{(0)}$.

Garg, Gentry and Halevi [24] recently proposed a candidate construction of a randomized graded encoding system, which has some additional algorithms to deal with the fact that the encoding is randomized. Another candidate was also proposed by Coron, Lepoint and Tibouchi [21]. We refer to these works for a precise definition of graded encodings. Here we note that graded encodings define an "approximate" version of a multilinear group family. For ease of exposition, we proceed our description of graded encodings by using the more abstract and simpler multilinear groups. Although graded encodings slightly depart from this abstraction (mainly because of the randomized "noisy" procedures), they can be adapted to work in place of multilinear groups.

In generic multilinear groups we assume the existence of an algorithm $\mathcal{G}(1^\lambda, k)$ that, on input the security parameter and an integer k indicating the number of levels (i.e., the number of allowed pairing operations), generates the description pp of leveled multilinear groups $(\mathbb{G}_1, \ldots, \mathbb{G}_k)$, each of large prime order $p > 2^\lambda$. We let g_i be a canonical generator of $\mathbb{G}_i$ and we assume that pp includes $g_1 \in \mathbb{G}_1$. The groups are such that there exists a set of bilinear maps $\{e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \rightarrow \mathbb{G}_{i+j}\}_{i,j \geq 1, i+j \leq k}$ such that $\forall a, b \in \mathbb{Z}_p$: $e_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab}$. When it is obvious from the context the indices i, j are dropped from $e_{i,j}$.

Limited Malleability. To prove the security of our scheme we assume that the encoding system is homomorphic *only* in a graded sense. Namely, given the level-1 encodings of the ℓ powers of $w \in R$, it must be hard to compute a level-k encoding of $w^{\ell k+1}$. Framed in the context of multilinear groups, this assumption can be seen as an extension of the computational Bilinear Diffie-Hellman Inversion assumption (first defined by Boneh, Boyen and Goh [9]) to the multilinear setting. Its hardness in the generic multilinear group follows by the same argument as in the "master theorem" of [9], i.e., by the linear independence of the polynomial $x^{\ell k+1}$ w.r.t. the polynomials $x, x^2, \ldots, x^{\ell k}$. It is worth noting that a similar assumption, in the multilinear groups setting, has been recently used by Hohenberger, Sahai and Waters [32].

Definition 3 ((ℓ, k)-Multilinear Diffie-Hellman Inversion). *Let* pp *be the description of a set of multilinear groups and* $g_1 \in \mathbb{G}_1$ *be a random generator. Let* $w \xleftarrow{\$} \mathbb{Z}_p$ *be chosen at random. We define the advantage of an adversary* $\mathcal{A}$ *in solving the* (ℓ, k)*-MDHI problem as* $\mathbf{Adv}_{\mathcal{A}}^{MDHI}(\lambda) = \Pr[\mathcal{A}(g_1, g_1^w, \ldots, g_1^{w^\ell}) = g_k^{w^{k\ell+1}}]$*, and we say that the* (ℓ, k)*-MDHI assumption holds for* $\mathcal{G}$ *if for every PPT* $\mathcal{A}$ *and for* $\ell = \mathsf{poly}(\lambda)$*,* $\mathbf{Adv}_{\mathcal{A}}^{MDHI}(\lambda)$ *is negligible in* λ*.*

The Scheme. Our homomorphic MAC based on k-linear groups allows for the following process: (1) one can generate constant-size tags, each authenticating the results of a computation $v_i = f_i(m_1, \ldots, m_n)$ for $i = 1$ to t, where each f_i is an arithmetic circuit of degree at most $D = \mathsf{poly}(\lambda)$; (2) one can compose the above computations f_i by using a "composition circuit" ϕ of degree at most k. Namely, one can finally authenticate $v = \phi(v_1, \ldots, v_t) = f^*(m_1, \ldots, m_n)$, where $f^* = \phi(f_1, \ldots, f_t)$.

Before describing our scheme in full detail, we first provide an intuitive description. The basic idea of our construction is to first generate the authentication tags as in [16] – i.e., as polynomials $y(X)$ – and to publish in the evaluation key level-1 encodings of the D powers of the secret value x, i.e., $h_i = g_1^{x^i}, i = 1, \ldots, D$. To authenticate a computation $v_i = f_i(m_1, \ldots, m_n)$, one first computes the polynomial $y^{(i)}(X) \leftarrow f_i(y_1(X), \ldots, y_n(X)) \in \mathbb{Z}_p[X]$, and then generates its "compressed" representation by computing the level-1 encoding $\Lambda_i = g_1^{y^{(i)}(x) - y^{(i)}(0)} = \prod_{j=1}^{d} (g_1^{x^j})^{y_j^{(i)}}$. To further authenticate the composed computation $\phi(v_1, \ldots, v_t)$ (with ϕ of degree k), the idea is to compute the k-level encoding $\Lambda = (g_k^{a^{k-1}})^{y(x)-y(0)}$, where $y(X) \in \mathbb{Z}_p[X]$ is the polynomial obtained from $\phi(y^{(1)}(X), \ldots, y^{(t)}(X))$. Precisely, Λ is computed by homomorphically evaluating ϕ over the level-1 encodings $\{\Lambda_i\}_i$: additions in ϕ are replaced by the group operation and multiplications are replaced by the pairing. In our scheme we also publish encodings $\{\eta_i = g_1^{ax^i} = h_i^a\}_{i=0}^{d}$ where $a \in \mathbb{Z}_p$ is randomly chosen, and we include in every tag another element $\Gamma = \Lambda^a$ computed by using the values $\{\eta_i\}_i$. Very roughly, these additional encodings are introduced to enable one to "move up" an encoding Λ_j from $\mathbb{G}_j$ to $\mathbb{G}_{j+1}$ without publishing the generator $g_1 \in \mathbb{G}_1$, i.e., one computes $\Lambda_{j+1} = e_{j,1}(\Lambda_j, g_1^a)$.

A full description of our scheme follows.

$\mathsf{KeyGen}(1^\lambda, D, k)$. Let λ be the security parameter, and D, k be two parameters of size $\mathsf{poly}(\lambda)$. The key generation works as follows.

Run $\mathcal{G}(1^\lambda, k)$ to generate the description of k-linear groups $\mathbb{G}_1, \ldots, \mathbb{G}_k$ of order p where p is a prime number of at least λ bits. Let $g_1 \xleftarrow{\$} \mathbb{G}_1$ be a random generator, and choose random values $a, x \xleftarrow{\$} \mathbb{Z}_p$, and a seed K of a pseudorandom function $F_K : \{0,1\}^* \to \mathbb{Z}_p$. Next, for $i = 1$ to D, compute $h_i = g_1^{x^i}, \eta_i = h_i^a$ and $A_1 = g_1^a$. Also, we let $g_i \in \mathbb{G}_i$ be the canonical generator of $\mathbb{G}_i$ which is obtained by repeatedly applying the graded map to g_1, i.e., let $g_2 = e(g_1, g_1)$ and $g_i = e(g_{i-1}, g_1)$. Similarly, we define A_i from A_1 and we observe that $A_i = g_i^{a^i}$. Finally, compute $\sigma_U = (1, (r_U - 1)/x)$ for a random $r_U \xleftarrow{\$} \mathbb{Z}_p$. σ_U is essentially a tag for the value 1 under a fixed canonical label (cf. the authentication algorithm).

Output $\mathsf{sk} = (K, g_1, x, a), \mathsf{ek} = (A_1, h_1, \eta_1, \ldots, h_D, \eta_D, \sigma_U)$ and let the message space $\mathcal{M}$ be $\mathbb{Z}_p$.

$\mathsf{Auth}(\mathsf{sk}, \tau, m)$. To authenticate a message $m \in \mathbb{Z}_p$ with label $\tau \in \{0,1\}^\lambda$, compute $r_\tau = F_K(\tau)$, set $y_0 = m$, $y_1 = (r_\tau - m)/x \bmod p$, and output $\sigma = (y_0, y_1) \in \mathbb{Z}_p^2$. The authentication tags produced by Auth are interpreted as degree-1 polynomials $y(X) = y_0 + y_1 X$ over the ring $\mathbb{Z}_p[X]$.

$\mathsf{Eval}_1(\mathsf{ek}, f, \boldsymbol{\sigma})$**.** This algorithm is the same as the Eval algorithm of the homomorphic MACs construction based on OWFs proposed in [16]: The input is the evaluation key ek, an arithmetic circuit $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$, and a vector $\boldsymbol{\sigma}$ of tags $(\sigma_1, \ldots, \sigma_n)$ where each σ_i is a polynomial $y^{(i)}(X) \in \mathbb{Z}_p[X]$. The authentication tag σ computed by Eval_1 is the polynomial $y(X)$ obtained by (homomorphically) evaluating the circuit f over the polynomial ring $\mathbb{Z}_p[X]$, i.e., $y(X) \leftarrow f(y^{(1)}(X), \ldots, y^{(n)}(X))$. Namely, additions/multiplications over $\mathbb{Z}_p$ are replaced by additions/multiplications of polynomials over $\mathbb{Z}_p[X]$.

$\mathsf{Compress}(\mathsf{ek}, \sigma)$**.** This algorithm takes as input an authentication tag σ of the form $y(X) \in \mathbb{Z}_p[X]$ of degree d (i.e., $y(X)$ consists of $d+1$ coefficients $(y_0, \ldots, y_d)$), and "compresses" the polynomial into a shorter value of constant size. The resulting tag is a triple $(y_0, \varLambda_1, \varGamma_1) \in \mathbb{Z}_p \times \mathbb{G}_1 \times \mathbb{G}_1$ where $\varLambda_1$ and $\varGamma_1$ are computed as follows: $\varLambda_1 = \prod_{i=1}^d h_i^{y_i}, \quad \varGamma_1 = \prod_{i=1}^d \eta_i^{y_i}$.

$\mathsf{Eval}_2(\mathsf{ek}, \phi, \boldsymbol{\sigma})$**.** This algorithm allows to further apply homomorphic operations on tags that were obtained using Eval_1 and later compressed using $\mathsf{Compress}$. Eval_2 takes as input the evaluation key ek, an arithmetic circuit $\phi : \mathbb{Z}_p^n \to \mathbb{Z}_p$ of degree at most k and a vector $\boldsymbol{\sigma}$ of tags $(\sigma_1, \ldots, \sigma_n)$ such that each σ_i is a triple $(y_0^{(i)}, \varLambda_1^{(i)}, \varGamma_1^{(i)}) \in \mathbb{Z}_p \times \mathbb{G}_1 \times \mathbb{G}_1$. Without loss of generality, we assume that in the circuit ϕ addition gates take inputs of the same degree i.[10]

Eval_2 evaluates the circuit ϕ over the tags by replacing additions and multiplications as follows:

- $\mathsf{Add}(\mathsf{ek}, \sigma_1, \sigma_2)$. On input two tags $\sigma_1 = (y_0^{(1)}, \varLambda_i^{(1)}, \varGamma_i^{(1)})$ and $\sigma_2 = (y_0^{(2)}, \varLambda_i^{(2)}, \varGamma_i^{(2)})$, it computes a tag $\sigma = (y_0, \varLambda_i, \varGamma_i)$ as follows: $y_0 = y_0^{(1)} + y_0^{(2)}, \quad \varLambda_i = \varLambda_i^{(1)} \cdot \varLambda_i^{(2)}, \quad \varGamma_i = \varGamma_i^{(1)} \cdot \varGamma_i^{(2)}$.
- $\mathsf{ConstMult}(\mathsf{ek}, \sigma_1, c)$. On input a tag $\sigma_1 = (y_0^{(1)}, \varLambda_i^{(1)}, \varGamma_i^{(1)})$ and a constant $c \in \mathbb{Z}_p$, it computes the tag $\sigma = (y_0, \varLambda_i, \varGamma_i)$: $y_0 = c \cdot y_0^{(1)}, \quad \varLambda_i = (\varLambda_i^{(1)})^c, \quad \varGamma_i = (\varGamma_i^{(1)})^c$.
- $\mathsf{Mult}(\mathsf{ek}, \sigma_1, \sigma_2)$. This takes as input two tags $\sigma_1 = (y_0^{(1)}, \varLambda_i^{(1)}, \varGamma_i^{(1)})$ and $\sigma_2 = (y_0^{(2)}, \varLambda_j^{(2)}, \varGamma_j^{(2)})$ and outputs a tag $\sigma = (y_0, \varLambda_d, \varGamma_d)$ where $d = i+j$. σ is computed as follows:

$$\begin{aligned}
y_0 &= y_0^{(1)} \cdot y_0^{(2)} \\
\varLambda_d &= e(\varLambda_i^{(1)}, \varGamma_j^{(2)}) \cdot e(\varLambda_i^{(1)}, A_j)^{y_0^{(2)}} \cdot e(A_i, \varLambda_j^{(2)})^{y_0^{(1)}} \\
\varGamma_d &= e(\varGamma_i^{(1)}, \varGamma_j^{(2)}) \cdot e(\varGamma_i^{(1)}, A_j)^{y_0^{(2)}} \cdot e(A_i, \varGamma_j^{(2)})^{y_0^{(1)}}
\end{aligned}$$

$\mathsf{Ver}(\mathsf{sk}, m, \mathcal{P}, \sigma)$**.** Let $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ be a labeled program, $m \in \mathbb{Z}_p$ and σ be a tag of either the form $(y_0, y_1) \in \mathbb{Z}_p^2$, or $(y_0, \varLambda_i, \varGamma_i) \in \mathbb{Z}_p \times \mathbb{G}_i^2$.

First, compute $\rho = f(r_{\tau_1}, \ldots, r_{\tau_n})$ where $r_{\tau_i} \leftarrow F_K(\tau_i)$. Next, according to the form of σ perform the following checks:

1. If $\sigma = (y_0, y_1) \in \mathbb{Z}_p^2$, then output 1 only if $\rho = y_0 + y_1 \cdot x \ \wedge \ y_0 = m$.

[10] Note that any circuit can be changed to meet this assumption: simply add multiplications by a special variable with value 1. This change does not increase the circuit's degree, and its homomorphic evaluation can be performed by using the tag σ_U.

2. If $\sigma = (y_0, \Lambda_i, \Gamma_i) \in \mathbb{Z}_p \times \mathbb{G}_1^2$, then output 1 only if $y_0 = m \quad \wedge \quad (g_i^{a^{i-1}})^{\rho - m} = \Lambda_i \ \wedge \ \Lambda_i^a = \Gamma_i$.

For lack of space, the correctness of the scheme is shown in the full version.

In the following theorem we prove the security of the scheme for the class of arithmetic circuits of (total) degree Δ such that $\Delta < p$, and in particular when $0 < \Delta/p < 1$ is the inverse of a small constant (e.g., 1/2). For lack of space, the proof of the theorem appears in the full version.

Theorem 2. *If F is a PRF and the computational (D, k)-MDHI assumption holds for $\mathcal{G}$, then the homomorphic MAC scheme described in Section 4 is secure.*

Possible Candidates. Here we discuss the possible instantiations of our scheme. A brief summary is also provided in Table 1.

GGH Graded Encodings. A first instantiation is obtained by using the recent proposal of multilinear maps [24, 21]. Since these realizations allow for a number of levels k which is polynomial in the security parameter, we obtain a homomorphic MAC that supports circuits of bounded polynomial degree and that, in particular, allows for degree-k composition. Also, due to the properties of the current multilinear maps realizations, the size of the final authentication tags (i.e., as generated by Eval_2) is $O(d)$ where $d \leq k$ is the degree of the composition circuit ϕ. This limitation stems from the fact that in all current realizations the size of an encoding at level d is $O(d)$. Hence, we obtain the following corollary.

Corollary 1. *Assume that F is a PRF, $\mathcal{G}$ is an instantiation of multilinear maps as in [24, 21], and the computational (D, k)-MDHI assumption holds for $\mathcal{G}$ with $D, k = \mathsf{poly}(\lambda)$. Then the scheme of Section 4 is a secure homomorphic MAC with authentication tags of size $O(k)$ and that supports computations expressed by arithmetic circuits of degree at most D and composition circuits of degree at most k.*

Supporting circuits of polynomial depth via compact multilinear maps. We note that the succinctness and the expressiveness (i.e., the class of circuits that are supported) of our construction crucially depend on the properties of the graded encoding. In particular, it is interesting to note that, in principle, we could support almost arbitrary circuits (i.e., of polynomial depth) and achieve full succinctness if the scheme is implemented with multilinear maps that allow for an exponential number of levels and that are compact. In this case, it is not even necessary to distinguish between Eval_1 and Eval_2: we can "merge" the algorithms Auth and $\mathsf{Compress}$ in order to create tags that are directly level-1 encodings, and then use Eval_2 to perform all the homomorphic operations. Although multilinear groups with such properties are not known, our result has the potential of yielding a fully-fledged homomorphic MAC. Indeed, our construction uses multilinear maps in a generic way, and its security holds against adversaries making an unbounded number of verification queries, in contrast to the fully-homomorphic MAC of Gennaro-Wichs, that can support only a constant number of verification queries.

Corollary 2. *Assume that F is a PRF, $\mathcal{G}$ is an (ideal) instantiation of compact multilinear maps, and the computational $(1, k)$-MDHI assumption holds for $\mathcal{G}$ for any $k < p/2 \approx 2^{\lambda-1}$. Then the scheme of Section 4 is a secure homomorphic MAC with authentication tags of size $O(1)$ and that supports computations expressed by arithmetic circuits of degree at most k.*

Scheme	**Tag Size**	**Composability**	**Supported Computations**	**Assumption**	**Verif. Queries**
CF13-1 [16]	$O(d)$	✓	degree-d circuits	OWF	✓
CF13-2 [16]	$O(1)$	×	degree-D circuits	D-DHI	✓
GW13 [28]	$O(\lambda)$	✓	Arbitrary circuits	FHE	×
This work with graded encodings [24, 21]	$O(k)$	degree-k circuits	degree-$(D+k)$ circuits	(D, k)-MDHI	✓
This work with ideal k-linear maps	$O(1)$	✓	degree-k circuits $\forall k : k/p < 1/2$	$(1, k)$-MDHI	✓

Table 1. Summary of homomorphic MACs instantiations with message space $\mathbb{Z}_p$. The last column indicates whether unbounded verification queries are supported or not.

Acknowledgements. The research of Dario Fiore is partially supported by the European Commission's Seventh Framework Programme Marie Curie Cofund Action AMAROUT II (grant no. 291803). The research of Rosario Gennaro was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defense and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

1. S. Agrawal and D. Boneh. Homomorphic MACs: MAC-based integrity for network coding. In M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, editors, *ACNS 09*, volume 5536 of *LNCS*, pages 292–305. Springer, June 2009.
2. N. Attrapadung and B. Libert. Homomorphic network coding signatures in the standard model. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 17–34. Springer, Mar. 2011.
3. N. Attrapadung, B. Libert, and T. Peters. Efficient completely context-hiding quotable and linearly homomorphic signatures. In *PKC 2013*, volume 7778 of *LNCS*, pages 386–404. Springer, 2013.

4. M. Backes, D. Fiore, and R. M. Reischuk. Verifiable delegation of computation on outsourced data. In *2013 ACM Conference on Computer and Communication Security*. ACM Press, November 2013.
5. S. Benabbas, R. Gennaro, and Y. Vahlis. Verifiable delegation of computation over large datasets. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 111–131. Springer, Aug. 2011.
6. N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS '12: Proceedings of the 3rd Symposium on Innovations in Theoretical Computer Science*, 2012.
7. N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. STOC, 2013.
8. N. Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, and O. Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, pages 315–333, 2013.
9. D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, May 2005.
10. D. Boneh, D. Freeman, J. Katz, and B. Waters. Signing a linear subspace: Signature schemes for network coding. In S. Jarecki and G. Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 68–87. Springer, Mar. 2009.
11. D. Boneh and D. M. Freeman. Homomorphic signatures for polynomial functions. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 149–168. Springer, May 2011.
12. D. Boneh and D. M. Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 1–16. Springer, Mar. 2011.
13. D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In J. Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 325–341. Springer, Feb. 2005.
14. X. Boyen. The uber-assumption family (invited talk). In S. D. Galbraith and K. G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, Sept. 2008.
15. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In R. Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, Oct. 2011.
16. D. Catalano and D. Fiore. Practical homomorphic MACs for arithmetic circuits. In *EUROCRYPT 2013*, 2013.
17. D. Catalano, D. Fiore, R. Gennaro, and K. Vamvourellis. Algebraic (trapdoor) one way functions and their applications. In *TCC 2013*, volume 7785 of *LNCS*, pages 680–699. Springer, 2013.
18. D. Catalano, D. Fiore, and B. Warinschi. Adaptive pseudo-free groups and applications. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 207–223. Springer, May 2011.
19. D. Catalano, D. Fiore, and B. Warinschi. Efficient network coding signatures in the standard model. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 680–696. Springer, May 2012.
20. K.-M. Chung, Y. T. Kalai, F.-H. Liu, and R. Raz. Memory delegation. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 151–168. Springer, Aug. 2011.

21. J.-S. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. In *CRYPTO 2013*, 2013.
22. D. Fiore and R. Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *2012 ACM Conference on Computer and Communication Security.* ACM Press, October 2012.
23. D. M. Freeman. Improved security for linearly homomorphic signatures: A generic framework. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 697–714. Springer, May 2012.
24. S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices and applications. In *EUROCRYPT 2013*, 2013.
25. R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482. Springer, Aug. 2010.
26. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT 2013*, pages 626–645, 2013.
27. R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin. Secure network coding over the integers. In P. Q. Nguyen and D. Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 142–160. Springer, May 2010.
28. R. Gennaro and D. Wichs. Fully homomorphic message authenticators. In *ASIACRYPT 2013*, 2013. Also in Cryptology ePrint Archive, Report 2012/290.
29. C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.
30. C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In L. Fortnow and S. P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.
31. S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. In R. E. Ladner and C. Dwork, editors, *40th ACM STOC*, pages 113–122. ACM Press, May 2008.
32. S. Hohenberger, A. Sahai, and B. Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In *CRYPTO 2013*, 2013.
33. R. Johnson, D. Molnar, D. X. Song, and D. Wagner. Homomorphic signature schemes. In B. Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 244–262. Springer, Feb. 2002.
34. J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.
35. S. Micali. Cs proofs. In *35th FOCS*, Nov. 1994.
36. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, May 1999.
37. B. Parno, M. Raykova, and V. Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In R. Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 422–439. Springer, Mar. 2012.
38. A. Shpilka and A. Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.
39. P. Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In R. Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 1–18. Springer, Mar. 2008.