Flexible and Efficient Verifiable Computation on Encrypted Data

Alexandre Bois[1⋆], Ignacio Cascudo[2], Dario Fiore[2], and Dongwoo Kim[3⋆⋆]

[1] CentraleSupelec, University of Paris-Saclay, Gif-sur-Yvette, France
alexandre.bois-verdiere@student.ecp.fr
[2] IMDEA Software Institute, Madrid, Spain
{ignacio.cascudo, dario.fiore}@imdea.org
[3] Western Digital Research, Milpitas, USA
Dongwoo.Kim@wdc.com

Abstract. We consider the problem of verifiable and private delegation of computation [Gennaro et al. CRYPTO'10] in which a client stores private data on an untrusted server and asks the server to compute functions over this data. In this scenario we aim to achieve three main properties: the server should not learn information on inputs and outputs of the computation (privacy), the server cannot return wrong results without being caught (integrity), and the client can verify the correctness of the outputs faster than running the computation (efficiency). A known paradigm to solve this problem is to use a (non-private) verifiable computation (VC) to prove correctness of a homomorphic encryption (HE) evaluation on the ciphertexts. Despite the research advances in obtaining efficient VC and HE, using these two primitives together in this paradigm is concretely expensive. Recent work [Fiore et al. CCS'14, PKC'20] addressed this problem by designing specialized VC solutions that however require the HE scheme to work with very specific parameters; notably HE ciphertexts must be over $\mathbb{Z}_q$ for a large prime q.
In this work we propose a new solution that allows a flexible choice of HE parameters, while staying modular (based on the paradigm combining VC and HE) and efficient (the VC and the HE schemes are both executed at their best efficiency). At the core of our new protocol are new homomorphic hash functions for Galois rings. As an additional contribution we extend our results to support non-deterministic computations on encrypted data and an additional privacy property by which verifiers do not learn information on the inputs of the computation.

1 Introduction

We address the problem of verifiable computation on encrypted data. This problem arises in situations where a client wants to compute some function over private data on an untrusted machine (a cloud server for example) and is concerned about three issues. The first one is *efficiency*: the client wants to take advantage of the machine's computing power and to do many fewer operations than those

⋆ Work done while at IMDEA Software Institute.
⋆⋆ Work done while at IMDARC, Seoul National University, Korea.

needed to execute the computation. The second one is *privacy*—the client wants to keep the data hidden to the server—and the third one is *integrity*—the client wants to ensure that the results provided by the untrusted machine are correct.

If the goal is to solve privacy (and efficiency), then fully homomorphic encryption (FHE) is the answer. With FHE the server can receive data encrypted and compute any function on it. The first FHE scheme was proposed in 2009 by Gentry [Gen09], and since then we have several families of more efficient schemes, e.g., [BV11, BGV12, FV12, CKKS17, GSW13, DM15, CGGI16, CGGI17].

If the goal is to solve integrity (and efficiency), then the problem is in the scope of verifiable computation (VC) [GGP10]. In a nutshell, with a VC protocol the server can produce a proof about the correctness of a computation, and this proof can be checked by the client faster than recomputing the function. As of today, there exist several solutions to this problem based on different approaches, such as doubly-efficient interactive proofs [GKR08], FHE and garbled circuits [GGP10], functional/attribute-based encryption [PRV12, GKP+13], and succinct (interactive and non-interactive) arguments for NP, e.g., [Kil92, GGPR13, AHIV17, WTs+18, BCR+19].

When it comes to solving *both privacy and integrity* (while retaining efficiency), there exist fewer solutions. Gennaro et al. [GGP10] proposed a VC scheme with privacy based on combining garbled circuits and FHE, and Goldwasser et al. [GKP+13] proposed a VC scheme with privacy of inputs (but not outputs) based on succinct single-key functional encryption . Unfortunately, the concrete efficiency of these two solutions is not satisfactory, e.g. [GGP10] require the full FHE power, and [GKP+13] needs attribute-based encryption for expressive predicates and works for functions with single-bit outputs.

A third approach is that of Fiore et al. [FGP14] who proposed a generic construction of VC with privacy, obtained by combining an FHE scheme and a VC scheme: the basic idea is to use VC to prove the correctness of the FHE evaluations on ciphertexts. Efficiency-wise this approach is promising as it tries to reconcile the best of the two lines of work that individually address privacy (FHE) and integrity (VC) and that have advanced significantly in terms of efficiency.

The efficiency challenges of proving correctness of FHE evaluation. The instantiation of [FGP14] generic construction still faces two challenges related to efficiency:

1. When executing a function g, the VC scheme must prove *the FHE evaluation of* g, whose representation is typically much larger than g, as it acts over FHE ciphertexts.
2. The FHE ciphertext space may not match the message space natively supported by the VC scheme. Although in theory this is not an issue for general-purpose VCs, in practice it would require expensive conversions that can significantly affect the cost of generating the VC proof. For example, many succinct arguments work for arithmetic circuits over a field $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$ where p is a large prime (e.g., the order of bilinear groups), whereas the FHE ci-

phertext spaces may be polynomial rings $\mathbb{Z}_q[X]/(f)$, $f \in \mathbb{Z}_q[X]$, where q is not necessarily a prime of exponential size (in the security parameter).

Fiore et al. [FGP14] proposed a concrete instantiation of their generic construction that, though supporting only the evaluation of quadratic functions, addresses both the two challenges above as follows. First, they use the HE scheme from [BV11], where a ciphertext consists of two polynomials in $\mathbb{Z}_q[X]/(f)$, and "force it" to work with $q = p$, where p is the prime order of bilinear groups used by their VC scheme. Second, they reduce the dependency of their VC scheme on the size of the ciphertexts (i.e., the degree d_f of the polynomial f) via a technique called homomorphic hashing. When executing a function g on n inputs, a strawman solution would require the VC to prove g's homomorphic evaluation, of size more than $O(d_f \cdot |g|)$. By using homomorphic hashing, they can have proofs generated in time $O(d_f \cdot n + |g|)$. Essentially, the ciphertext size impacts the cost of proof generation only on the number of inputs, which is unavoidable.

Recently, [FNP20] extended the approach of [FGP14] to support public verifiability and the evaluation of more than quadratic functions (still of constant degree) via the use of specialized zkSNARKs for polynomial rings. However, the scheme of [FNP20] still requires to instantiate the [BV11] HE scheme with a specific q order of the bilinear groups used by the zkSNARK.

To summarize, existing solutions [FGP14, FNP20] manage to avoid expensive conversions and achieve efficient proof generation, but pay the price of imposing specific values to the parameters of the HE scheme. This choice has several drawbacks. One problem is the efficiency of the HE scheme: the size of the modulus q mainly depends on the complexity of the computations to be supported (for correctness) and there are many cases where it can be as small as 50–60 bits and not necessarily a prime. Forcing it to be a prime of, say, 256 bits (because of discrete log security in bilinear groups) not only makes it unreasonably larger but also requires, due to security of the RingLWE problem, to increase the size of the polynomial ring, i.e., the degree of f. Similarly, in cases where for HE correctness q would be larger than 256 bits, then one must instantiate the bilinear groups at a security level higher than necessary. So, all in all, the techniques of [FGP14, FNP20] do not allow flexible choices of HE parameters.

1.1 Our Contributions

In this paper we provide new VC schemes with privacy of inputs and outputs that solve the two aforementioned efficiency challenges while staying *modular* (based on VC and FHE used independently) and *flexible* (no need to tweak the HE parameters).

Our VC schemes support HE computations of constant multiplicative depth, offer public delegation and public verifiability, and are multi-function, i.e., one can encode inputs with a function-independent key and delegate the execution of multiple functions on these inputs (see [PRV12]). These features are similar to those of the recent work [FNP20].

In contrast to previous works [FGP14, FNP20], we can use the [BV11] somewhat homomorphic encryption scheme (where ciphertexts are in $\mathbb{Z}_q[X]/(f)$) *in-*

Scheme	Delegation	Verification	Max degree	HE modulus q
[FGP14]	priv	priv	2	prime $> 2^\lambda$
[FNP20]	pub	pub	const	prime $> 2^\lambda$
Ours	pub	pub	const	any

Table 1. Comparison of efficient VC schemes with privacy of inputs/outputs based on homomorphic encryption.

stantiated with any choice of the ciphertext modulus q.[4] This flexibility enables better and faster instantiations of the HE component than in [FGP14, FNP20].

For instance, in applications where q can be of about 50 bits we can set $deg(f) = 2^{11}$, which makes ciphertexts $40\times$ shorter than using a 250-bits prime q, which would require $deg(f) = 2^{14}$. Furthermore, the fact that our modulus q does not have to be prime may lead to use optimized circuits and thus to faster executions in practice (an example is the lowest-digit-removal polynomial in [CH18], that has a lower degree for a modulus p^e than for a close prime modulus).

As a key technique to achieve flexibility and to gain efficiency by working on smaller spaces, we define and construct new, more general, homomorphic hash functions for Galois rings. Briefly speaking, these functions can compress a ciphertext element from a polynomial ring $\mathbb{Z}_q[X]/(f)$, where $q = p^e$ for a prime p and f is arbitrary, into a smaller Galois ring (i.e., a polynomial ring $\mathbb{Z}_q[X]/(h)$ such that h is monic and its reduction modulo p is irreducible in $\mathbb{Z}_p[X]$). Next, thanks to the homomorphic property, we can use any VC scheme for proving arithmetic circuits over $\mathbb{Z}_q[X]/(h)$. As a concrete example, we show how to use the efficient GKR protocol [GKR08] for this task. In terms of efficiency, $h \in \mathbb{Z}_q[X]$ is a polynomial whose degree governs the soundness of the proofs (we need that $1/p^{deg(h)}$ is negligible) and is concretely smaller than the degree of f, e.g., $deg(h)$ can be between $2^4\times$ (when $p = 2$) and $2^{11}\times$ (when $p = q$) smaller.

We stress that previous homomorphic hash functions from [FGP14, FNP20] only map from $\mathbb{Z}_q[X]/(f)$ to $\mathbb{Z}_q$ and need q be a large prime. So they would not allow flexible choices of parameters. Our constructions instead have no restriction on q and can fine-tune the output space according to the desired soundness.

At the core of our result is a technique to speed up the prover costs in verifiable computation over polynomial rings. Given that polynomial rings are common algebraic structures used in many lattice-based cryptographic schemes, our methods and analysis might be easily reusable in other contexts different from FHE. As an example, in Section 3.2 we show how our technique can be used to obtain a verifiable computation scheme for parallel, single-instruction multiple-data (SIMD) computations where the proof generation costs are minimally dependent on the number of parallel inputs.

[4] Precisely, our basic scheme in Section 3 works for a q that is a prime power; in the full version of this paper we generalize it to any (possibly composite) integer q.

Extensions for non-deterministic computations and context-hiding. As an additional contribution, we generalize the notion of private VC to support non-deterministic computations along with context-hiding.

In brief, supporting nondeterministic computations means to consider functions of the form $g(x, w)$ in which the untrusted worker receives an encryption of the input x, holds an additional input w and can produce an encoding of $g(x, w)$. In this case, the security property becomes analogous to the one we have in proof systems for NP, namely the untrusted worker can produce an encoding y that is accepted only if there exists a w such that $y = g(x, w)$. Nondeterminism is useful to handle more complex computations, such as ones that use random coins, non-arithmetic operations (e.g., bit-decompositions) or (secret) computation parameters. For instance, with this we can prove re-randomization of ciphertexts, or evaluate polynomials with coefficients provided by the server.

To provide privacy against verifiers, we consider the notion of context hiding of [FNP20], which guarantees that the verifier learns no information about the input x beyond what can be inferred from the output $y = g(x)$. In our work, we extend context-hiding to the non-deterministic setting to ensure that the verifier learns nothing about (x, w). This includes both verifiers that only receive computation's results, and those who generated the input and the corresponding ciphertext/encoding (in which case x is already known).

Next, we extend our flexible VC constructions to support non-deterministic computations and achieve context-hiding. In particular, we show a scheme that is based on proving correctness of [BV11] HE evaluations and in which we address the two efficiency challenges mentioned earlier using our homomorphic hashing technique, thanks to which we keep the cost of proof generation $O(d_f \cdot n + |g|)$. To achieve context-hiding, however, instead of a verifiable computation for arithmetic circuits over the Galois ring $\mathbb{Z}_q[X]/(h)$, we use a commit-and-prove succinct zero-knowledge argument for circuits over this Galois ring. The latter could be instantiated by using existing schemes for $\mathbb{Z}_p$ (recall p is the prime such that $q = p^e$). The design of efficient ZK arguments that can directly and efficiently handle Galois rings is an interesting open problem for future research.

1.2 Organization

In Section 2, we introduce notation and preliminary definitions. Section 3 presents our generic VC scheme on encrypted data. In Section 4, we discuss an instantiation of our VC scheme and present our homomorphic hash functions. In Section 5, we further develop our scheme to handle nondeterministic computations with context-hiding.

2 Preliminary Definitions

In this section, we recall notation and basic definitions. Some of them are recalled only informally; we refer to the full version for more formal definitions.

Notation. Let $\lambda \in \mathbb{N}$ be the security parameter. We say that a function F is *negligible* in λ and denote it by $\mathsf{negl}(\lambda)$ if $F(\lambda) = o(\lambda^{-c})$ for all $c > 0$. A

probability is said to be *overwhelming* if it is $1 - \mathsf{negl}(\lambda)$. Let $\mathcal{D}$ be a probability distribution and S be a set. The notation $r \xleftarrow{\$} \mathcal{D}$ means that r is randomly sampled from the distribution $\mathcal{D}$, while $r \xleftarrow{\$} S$ means that r is sampled uniformly randomly from the set S. All adversaries $\mathcal{A}$ and entities (a prover $\mathcal{P}$ and a verifier $\mathcal{V}$) in this paper are probabilistic polynomial-time (PPT) Turing machines. In this paper, a ring is always a commutative ring with a multiplicative identity 1.

2.1 Verifiable Computation

We recall the definition of a verifiable computation (VC) scheme [GGP10]. We use the notion of Multi-Function VC scheme from [PRV12], with a slight modification to handle public delegatability and verifiability (we will simply call this a VC scheme in the rest of this work). A multi-function VC scheme allows the computation of several functions on a single input and satisfies an adaptive security notion where the adversary can see many input encodings before choosing the function (similarly as the definition of a Split Scheme in [FGP14]).

Definition 1 (Verifiable Computation). *A* verifiable computation scheme $\mathcal{VC}$ *consists of a tuple of algorithms* (Setup, KeyGen, ProbGen, Compute, Verify, Decode)*:*

$\mathsf{Setup}(1^\lambda) \to (PK, SK)$: *produces the public and private parameters that do not depend on the functions to be evaluated.*

$\mathsf{KeyGen}_{PK}(g) \to (PK_g, SK_g)$: *produces a keypair for evaluating a specific function g.*

$\mathsf{ProbGen}_{PK}(x) \to (\sigma_x, \tau_x)$: *The problem generation algorithm uses the public key PK to encode the input x as a value σ_x that is given to the server to compute with, and a public value τ_x which is given to the verifier.*

$\mathsf{Compute}_{PK_g}(\sigma_x) \to \sigma_y$: *Using a public key for a function g and the encoded input σ_x, the server computes an encoded version σ_y of the function's output $y = g(x)$.*

$\mathsf{Verify}_{PK_g}(\tau_x, \sigma_y) \to acc$: *Using the public key for a function g, and a verification token τ_x for an input x, the verification algorithm converts the server's output σ_y into a bit acc. If $acc = 1$ we say the client accepts (which means that σ_y decodes to $y = g(x)$ – see below), otherwise if $acc = 0$ we say the client rejects.*

$\mathsf{Decode}_{SK,SK_g}(\sigma_y) \to y$: *using the secret keys, this algorithm decodes an output encoding σ_y to some value y.*

Remark 1. In our definition we did not include PK among the inputs of Compute and Verify; this can be done without loss of generality as in any scheme one can include PK into PK_g. Also, note that ProbGen takes only PK (and not PK_g) as an input; this highlights the fact that inputs can be encoded independently of the functions g that will be executed on them. Finally, we could have included SK among the inputs of KeyGen; in this case, however, one would partially lose the public delegation property.

A VC scheme satisfies *correctness*, *security*, *privacy*, and *outsourceability* whose definition is as follows:

Correctness. For any function g and input x,

$$\Pr\left[\begin{array}{c|r} \mathsf{Verify}_{PK_g}(\tau_x,\sigma_y)=1 & (PK,SK)\leftarrow\mathsf{Setup}(1^\lambda) \\ \wedge\ \mathsf{Decode}_{SK,SK_g}(\sigma_y)=g(x) & (PK_g,SK_g)\leftarrow\mathsf{KeyGen}_{PK}(g) \\ & (\sigma_x,\tau_x)\leftarrow\mathsf{ProbGen}_{PK}(x) \\ & \sigma_y\leftarrow\mathsf{Compute}_{PK_g}(\sigma_x) \end{array}\right]=1.$$

To define security and privacy, we first describe the following experiments:

Experiment $\mathbf{Exp}_{\mathcal{A}}^{Verif}[\mathcal{VC},\lambda]$
$(PK,SK)\leftarrow\mathsf{Setup}(1^\lambda)$;
$(x,st)\leftarrow\mathcal{A}^{O_{\mathsf{KeyGen}}(\cdot)}(PK)$;
$(\sigma_x,\tau_x)\leftarrow\mathsf{ProbGen}_{PK}(x)$;
$(g,\hat{\sigma}_y)\leftarrow\mathcal{A}^{O_{\mathsf{KeyGen}}(\cdot)}(st,\sigma_x,\tau_x)$;
$acc\leftarrow\mathsf{Verify}_{PK_g}(\tau_x,\hat{\sigma}_y)$;
if $acc=1$ and $\mathsf{Decode}_{SK,SK_g}(\hat{\sigma}_y)\neq g(x)$
output 1;
else output 0;

Experiment $\mathbf{Exp}_{\mathcal{A}}^{Priv}[\mathcal{VC},g,\lambda]$
$b\leftarrow\{0,1\}$;
$(PK,SK)\leftarrow\mathsf{Setup}(1^\lambda)$;
$(x_0,x_1,st)\leftarrow\mathcal{A}^{O_{\mathsf{KeyGen}}(\cdot)}(PK)$;
$(\sigma_b,\tau_b)\leftarrow\mathsf{ProbGen}_{PK}(x_b)$;
$\hat{b}\leftarrow\mathcal{A}^{O_{\mathsf{KeyGen}}(\cdot)}(st,\sigma_b,\tau_b)$;
if $b=\hat{b}$ output 1;
else output 0;

In the experiments above, $O_{\mathsf{KeyGen}}(g)$ is an oracle that can be called only once, it runs $PK_g\leftarrow\mathsf{KeyGen}_{PK}(g)$ and returns PK_g. The one-time use of the oracle is done for simplicity. Indeed, consider an experiment in which the adversary is allowed to query this oracle multiple times: an adversary playing in such an experiment can be reduced to one playing in the experiment above in a straightforward way, as the KeyGen algorithm uses only a public key and thus can be easily simulated. Similarly, this is why it is enough to give to the adversary only one specific encoding using ProbGen.

Security. For any PPT adversary $\mathcal{A}$, $\Pr[\mathbf{Exp}_{\mathcal{A}}^{Verif}[\mathcal{VC},\lambda]=1]\leq\mathsf{negl}(\lambda)$. Note that this is an adaptive notion of security, as defined in [FGP14].

Privacy. For any PPT adversary $\mathcal{A}$ and for any function g,

$$\Pr[\mathbf{Exp}_{\mathcal{A}}^{Priv}[\mathcal{VC},g,\lambda]=1]\leq\frac{1}{2}+\mathsf{negl}(\lambda).$$

Remark 2. Our definition of verifiable computation has public verifiability (anyone can use Verify only with public keys and τ_x) and public delegatability (anyone can run ProbGen and KeyGen). This immediately implies the notion of privacy in the presence of verification queries given in [FGP14], namely the scheme stays private even if the adversary learns whether its results are accepted or not.

Outsourceability. For any x and any honestly produced σ_y, the time required for $\mathsf{ProbGen}_{PK}(x)$, $\mathsf{Verify}_{PK_g}(\tau_x,\sigma_y)$, and $\mathsf{Decode}_{SK,SK_g}(\sigma_y)$ is $o(T)$ where T is the time required to compute $g(x)$, i.e., it allows efficient problem generation and verification followed by decoding.

The VC constructions we present in this paper are first built as public-coin interactive protocols which can be made non-interactive using the Fiat-Shamir heuristic. Therefore, we also consider interactive versions of Compute and Verify. Also, our constructions work in a simpler model in which KeyGen only outputs a public key without SK_g.

2.2 Fully Homomorphic Encryption

We briefly recall the notion of *(public-key) fully homomorphic encryption scheme* (FHE), which is is a tuple of algorithms ($\mathsf{FHE.ParamGen}, \mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval}$) working as follows:

$\mathsf{FHE.ParamGen}(1^\lambda)$: generates the public parameters (e.g., description of plaintext space $\mathcal{M}$, ciphertext space, key space, randomness distributions, etc.) which are assumed input to all subsequent algorithms.
$\mathsf{FHE.KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{evk}, \mathsf{dk})$: outputs a public encryption key pk, a public evaluation key evk, and a secret decryption key dk.
$\mathsf{FHE.Enc}_{\mathsf{pk}}(m) \to c$: encrypts a message $m \in \mathcal{M}$, and outputs ciphertext c.
$\mathsf{FHE.Dec}_{\mathsf{dk}}(c) \to m^*$: decrypts a ciphertext c into a plaintext $m^* \in \mathcal{M}$.
$\mathsf{FHE.Eval}_{\mathsf{evk}}(g, c_1, \ldots, c_n) \to c^*$: Given the evaluation key evk, a circuit $g : \mathcal{M}^n \to \mathcal{M}$, and n ciphertexts $c_1, \ldots, c_n$, it computes an output ciphertext c^*.

An FHE scheme should satisfy the usual notion of *correctness* and *semantic security*. In addition, we say that FHE is *compact* if the ciphertext size is bounded by some fixed polynomial in the security parameter, and is independent of the size of the evaluated circuit or the number of inputs it takes. We refer to the full version for the formal definitions.

In our work, we mainly consider SHE (a.k.a. somewhat homomorphic encryption), a restricted FHE notion that guarantees above-mentioned properties only if the circuit g is of a bounded degree which is fixed a-priori in $\mathsf{SHE.ParamGen}$.

2.3 Succinct Argument Systems

Let $\mathcal{R}$ be an NP relation. An argument system Π for $\mathcal{R}$ comprises three algorithms $(\Pi.\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ working as follows:

$\Pi.\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$: outputs a common reference string crs.
$\mathcal{P}(\mathsf{crs}, x, w)$: given a statement x and a witness w, the prover interacts with the verifier below.
$\mathcal{V}(\mathsf{crs}, x)$: given a statement x, the verifier outputs 0 (reject) or 1 (accept) after interacting with the prover.

We denote by $\langle \mathcal{P}(\mathsf{crs}, x, w), \mathcal{V}(\mathsf{crs}, x) \rangle_\Pi = b$, with $b \in \{0, 1\}$, an execution between $\mathcal{P}$ and $\mathcal{V}$ where b is $\mathcal{V}$'s output at the end of the interaction. If $\mathcal{V}$ uses public randomness only, Π is said *public-coin*. Π is said *succinct* if the protocol's communication is at most polylogarithmic in the witness size. An argument system Π satisfies the standard *completeness* and *soundness* properties (see full version for the formal definitions).

3 Our VC Scheme - Generic Solution

In this section, we present our generic VC scheme for private verifiable computation. The high-level idea is to apply a succinct argument system on the image of evaluation process $(\mathsf{SHE.Eval}_{\mathsf{evk}}(g, c_1, \ldots, c_t))$ of SHE under a homomorphic hash function.

3.1 Building Blocks and Assumptions

Our generic VC scheme consists of three building blocks: SHE, Homomorphic Hash Functions, and a Succinct Argument System for deterministic polynomial-time computations. We first describe the assumptions on each building block necessary for the construction of the generic VC scheme. It will be shown, in Section 4, that these assumptions can be met to provide an instantiation of the VC scheme.

Notation. Let $R = \mathbb{Z}[X]/(f)$ denote a quotient polynomial ring with $f \in \mathbb{Z}[X]$, a monic polynomial of degree d_f. For a positive integer t, $R_t := R/tR = \mathbb{Z}_t[X]/(f)$. We use q to denote a power of some prime p, i.e., $q = p^e$.

Somewhat Homomorphic Encryption. We assume that the ciphertext space of given SHE is $R_q^D = (\mathbb{Z}_q[X]/(f))^D$ where $q = p^e$ is a power of prime p, and D is a positive integer. We also assume that the evaluation algorithm $\mathsf{SHE.Eval}_{\mathsf{evk}}$ can be represented by an arithmetic circuit[5] over the ring R_q (or R_q^D).

Homomorphic Hash Functions. To gain efficiency in proving (and verification), we exploit a homomorphic hash function defined by a ring homomorphism $H : \mathbb{Z}_q[X]^D \to \mathcal{D}_H$ to a ring $\mathcal{D}_H$. Let $\mathcal{H}$ be a family of hash functions $\{H\}$ where each H is as described above and the uniform sampling of $H \in \mathcal{H}$ can be done with a public-coin process. We assume that $\mathcal{H}$, when the domain is restricted to a subset $\mathcal{D} \subset \mathbb{Z}_q[X]^D$, is ε-universal whose definition follows.

Definition 2 (ε-Universal Hash Functions). *A family $\mathcal{H}$ of hash functions is ε-universal if for all $c, c' \in \mathcal{D}$ such that $c \neq c'$, it holds that*

$$\Pr[H \xleftarrow{\$} \mathcal{H} : H(c) = H(c')] \leq \varepsilon.$$

We additionally assume that the set $\mathcal{D}$ in the above definition is large enough so that all ciphertexts arising can be embedded into it.

Succinct Argument System. We assume a public-coin succinct argument system Π that works for the relations represented by a (polynomial-size) arithmetic circuit over the rings $\mathcal{D}_H$ for all $H \in \mathcal{H}$.

3.2 The Generic Scheme

We now give a description of the generic private VC scheme using the building blocks and notation from Section 3.1. Our scheme follows the VC syntax from Section 2.1, except for $\mathsf{Compute}$ and Verify that we describe as a public-coin interactive protocol for two main reasons. First, we make use of a succinct argument system that can be interactive. Second, for security reasons, a homomorphic hash function must be sampled uniformly at random, unpredictably by a prover, e.g., a verifier samples and notifies a homomorphic hash function *after* a prover claimed an output. Note that a non-interactive version of our VC can be obtained in the random oracle model by applying the Fiat-Shamir transform.

[5] It is composed of gates performing addition or multiplication.

In our VC scheme, a verifier $\mathcal{V}$ encrypts the input $x = (x_1, x_2, \ldots, x_n)$ (with SHE) and sends the encrypted inputs $(c_i)_{i=1}^n = (\mathsf{SHE.Enc}(x_i))_{i=1}^n \in (R_q^D)^n = ((\mathbb{Z}_q[X]/(f))^D)^n$ to a prover $\mathcal{P}$. We remark that $\mathcal{P}$ performs the homomorphic evaluation $\mathsf{SHE.Eval}_{\mathsf{evk}}(g, c_1, \ldots, c_n)$ *without* reduction modulo f, and then proves this computation. Namely, $\mathcal{P}$ computes the function $\hat{g} : (R_q^D)^n \to \mathbb{Z}_q[X]^D$ (not R_q^D) that describes $\mathsf{SHE.Eval}_{\mathsf{evk}}(g, \cdot)$ without reduction modulo f. In other words, $\hat{g}$ is such that:

$$\hat{g}(c_1, ..., c_n) \mod f = \mathsf{SHE.Eval}_{\mathsf{evk}}(g, c_1, ..., c_n) \in R_q^D.$$

The VC scheme consists of a tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{ProbGen}, \mathsf{Compute}, \mathsf{Verify}, \mathsf{Decode})$ as follows.

$\mathsf{Setup}(1^\lambda) \to (PK, SK)$:
- Run $(\mathsf{pk}, \mathsf{evk}, \mathsf{sk}) \leftarrow \mathsf{SHE.KeyGen}(1^\lambda)$ to generate keys for SHE.
- Set $PK = (\mathsf{pk}, \mathsf{evk})$ and $SK = \mathsf{sk}$.

$\mathsf{KeyGen}_{PK}(g) \to (PK_g, SK_g)$:
- Run $\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$ to generate the common reference string of Π for the circuit $\hat{g} : (R_q^D)^n \to \mathbb{Z}_q[X]^D$ over the ciphertexts.
- Set $PK_g = (PK, \hat{g}, \mathsf{crs})$ and $SK_g = \emptyset$.

$\mathsf{ProbGen}_{PK}(x) \to (\sigma_x, \tau_x)$:
- Parse x as $(x_1, x_2, \ldots, x_n)$.
- Run $c_i \leftarrow \mathsf{SHE.Enc}_{\mathsf{pk}}(x_i)$ for $i \in \{1, 2, \ldots, n\}$ to get ciphertexts $c_x = (c_1, c_2, \ldots, c_n) \in \left(R_q^D\right)^n$.
- Set $\sigma_x = \tau_x = c_x$.

$\langle \mathsf{Compute}_{PK_g}(\sigma_x), \mathsf{Verify}_{PK_g}(\tau_x) \rangle$: prover and verifier proceed as follows.
- $\mathsf{Compute}$ computes $c_y = \hat{g}(c_x)$ and sends it to Verify.
- Verify samples and sends a homomorphic hash function $H \xleftarrow{\$} \mathcal{H}$.
- $\mathsf{Compute}$ and Verify both compute $\gamma_1 = H(c_1), \ldots, \gamma_n = H(c_n), \gamma_y = H(c_y)$.
- $\mathsf{Compute}$ and Verify run the argument system $\langle \mathcal{P}(\mathsf{crs}, \hat{g}, \gamma_1, \ldots, \gamma_n, \gamma_y), \mathcal{V}(\mathsf{crs}, \hat{g}, \gamma_1, \ldots, \gamma_n, \gamma_y) \rangle_\Pi$ in the roles of prover and verifier respectively. This is for $\mathcal{P}$ to convince $\mathcal{V}$ that $\gamma_y = \hat{g}(\gamma_1, \ldots, \gamma_n)$ over the ring $\mathcal{D}_H$.
- Let σ_y include c_y and the transcript of the interactive argument.
- Let b be the bit returned by $\mathcal{V}$. Verify accepts if and only if $b = 1$.

$\mathsf{Decode}_{SK}(\sigma_y) \to y$: Compute $y = \mathsf{SHE.Dec}_{\mathsf{sk}}(c_y \ mod \ f)$.

Notice that if Π is a k-round public-coin interactive protocol, then the VC protocol described above is a $(k+1)$-round public-coin protocol. By applying Fiat-Shamir to such a protocol, we obtain a non-interactive VC scheme in which the random oracle is used to derive the homomorphic hash function H and all the random challenges of $\mathcal{V}$ in Π. Then, $\mathsf{Compute}$ can be described as a non-interactive algorithm that on input σ_x outputs $\sigma_y = (c_y, \pi)$ where π are all the messages of $\mathcal{P}$, while $\mathsf{Verify}(\tau_x, \sigma_y)$ is the algorithm that returns the acceptance bit of the non-interactive verifier on the hashed inputs, i.e., $\mathcal{V}(PK_g, H(c_1), \ldots, H(c_n), H(c_y))$.

The application of Fiat-Shamir may incur a security loss, which mainly boils down to its application to the k-round Π protocol. As shown in [BCS16], if Π satisfies the notion of state-restoration soundness this loss is only polynomial in the number of rounds. Notably, Canetti et al. [CCH+18] proved that the GKR protocol (that we consider in Section 4 to instantiate Π) satisfies this property.

Remark 3 (On a variant using universal arguments). Note that, if Π is a preprocessing argument system with a *universal* CRS (i.e., following the notion in [CHM+20]), then one can modify our VC scheme as follows: Π.Setup can be executed in Setup (of VC scheme) and the universal CRS is included in PK, while KeyGen would only run the deterministic preprocessing $\mathsf{crs}_g \leftarrow \mathsf{Preprocess}(\mathrm{CRS}, g)$. The main benefit of this variant is that only the Setup algorithm must be executed in a trusted manner.

The generic scheme satisfies the properties of VC scheme given that all the building blocks satisfy the required properties.

Theorem 1. *For given security parameter λ, if we exploit a correct, compact, and secure* SHE *scheme, an ε-universal family of hash functions with $\varepsilon = \mathsf{negl}(\lambda)$, and a complete succinct argument system Π with soundness $\delta = \mathsf{negl}(\lambda)$, then our VC scheme is correct, secure, private and outsourceable.*

Proof. We refer to the formal definition of VC scheme (Definition 1) in Section 2.1. Our VC scheme is an interactive version of the generic private VC scheme from [FGP14], with the difference that there is an interactive Verify algorithm that uses homomorphic hashing. Therefore, we get the result as in [FGP14], except for the security for which we give a detailed proof.

The correctness of our VC scheme follows from the correctness of SHE and the completeness of the argument system Π.

The privacy of our VC scheme follows from the semantic security of SHE: if $\mathcal{P}$ could break the privacy of VC, it can run the VC scheme by itself on a ciphertext $\mathsf{SHE.Enc}(x_b)$ from the semantic security game of SHE then guess b with non-negligible advantage.

The outsourceability follows from the compactness of SHE scheme and the succinctness of the argument system Π that has verifier complexity $o(S)$ where S is the size of the circuit $\hat{g}$ (see next Lemma for a detailed complexity analysis).

For the security, we consider a slightly different version of the security experiment $\mathbf{Exp}_{\mathcal{A}}^{Verif}[\mathcal{VC}, \lambda]$ in Section 2.1, adapted to handle the case of protocols in which Compute and Verify interact. Namely, instead of an adversary that directly provides the result σ_y, we consider an adversary $\mathcal{A}$ that interacts with the challenger acting as an honest verifier, and $\mathcal{A}$ wins if the challenger accepts but the transcript decodes to a wrong result.

Let x and g respectively be the input and function chosen by $\mathcal{A}$ in the game, c_x be the encryption of x sent to $\mathcal{A}$, $\hat{c}_y$ be the result claimed by $\mathcal{A}$ in the first round, and c_y be the true result. We note $y = g(x)$ (if $\hat{c}_y$ does not encrypt y then necessarily $\hat{c}_y \neq c_y$). We now study the Verify algorithm: a homomorphic hash function $H \in \mathcal{H}$ is randomly sampled, either by $\mathcal{V}$ or by a random oracle

using $\mathcal{A}$'s inputs and outputs, so that $\mathcal{A}$ cannot know it before sending $\hat{c}_y$. Thus, if we denote by A the event $\{H(\hat{c}_y) = H(c_y)\}$ and if $\mathcal{H}$ is a ε-universal family then $\Pr[A|c_y \neq \hat{c}_y] \leq \varepsilon$. If $H(\hat{c}_y) \neq H(c_y)$ then the only way left for $\mathcal{A}$ to have $\mathcal{V}$ accept is to cheat when applying Π with the false result $H(\hat{c}_y)$. Let B denote the event $\{\Pi(\mathcal{V}, \mathcal{A}, H(c), H(\hat{c}_y), r) = 1\}$. If Π has soundness δ then $\Pr\left[B|\bar{A} \cap (c_y \neq \hat{c}_y)\right] \leq \delta$.

The output bit b of the security game satisfies:

$$\begin{aligned}\Pr[b = 1] &= \Pr[(A \cup B) \cap (\mathsf{SHE.Dec}(\hat{c}_y \ mod \ f) \neq y)] \\ &\leq \Pr[(A \cup B)|c_y \neq \hat{c}_y] \\ &\leq \Pr\left[B|\bar{A} \cap (c_y \neq \hat{c}_y)\right] + \Pr[A|c_y \neq \hat{c}_y] \\ &\leq \delta + \varepsilon\end{aligned}$$

This proves the result when ε and δ are $\mathsf{negl}(\lambda)$. □

The required computational (or communication) cost of our VC scheme can be easily derived from that of the argument system Π and the homomorphic hash functions $\mathcal{H}$.

Lemma 1. *Let $T^{\Pi}_{\mathcal{P}}, T^{\Pi}_{\mathcal{V}}$, and C^{Π} respectively denote the time complexity of $\mathcal{P}$, that of $\mathcal{V}$, and the communication cost in the argument system Π, which will be signified as, e.g., $T^{\Pi}_{\mathcal{P}}(g; R)$ when denoting the complexity of $\mathcal{P}$ (in Π) for a circuit g over a ring R. Then, for a circuit g with n inputs and 1 output, the time complexity of $\mathsf{Compute}_{PK_g}(\sigma_x)$, that of $\mathsf{Verify}_{PK_g}(\tau_x)$, and the communication cost in the execution of $\langle \mathsf{Compute}_{PK_g}(\sigma_x), \mathsf{Verify}_{PK_g}(\tau_x) \rangle$, in our VC scheme (Theorem 1) is as follows:*

$$\begin{aligned} \textit{Time}\ [\mathsf{Compute}_{PK_g}(\sigma_x)] &: T_{\hat{g}} + (n+1) \cdot T_{Hash} + T^{\Pi}_{\mathcal{P}}(\hat{g}; \mathcal{D}_H) \\ \textit{Time}\ [\mathsf{Verify}_{PK_g}(\tau_x)] &: (n+1) \cdot T_{Hash} + T^{\Pi}_{\mathcal{V}}(\hat{g}; \mathcal{D}_H) \\ \textit{Comm}\ [\langle \mathsf{Compute}_{PK_g}(\sigma_x), \mathsf{Verify}_{PK_g}(\tau_x) \rangle] &: |\hat{g}(c)| + |H| + C^{\Pi}(\hat{g}; \mathcal{D}_H) \end{aligned}$$

where $\hat{g}$ is the circuit corresponding to g over the ciphertext, $\mathcal{D}_H$ is the range space (ring) of a hash function $H \in \mathcal{H}$, $T_{\hat{g}}$ and T_{Hash} are the times for computing $\hat{g}$ (without reduction modulo f) and for evaluating a homomorphic hash function, respectively, $|\hat{g}(c)|$ is the size of the output ciphertext (from $\mathcal{P}$), and $|H|$ is the size of a homomorphic hash function.

The proof of this lemma directly follows from the description of the VC scheme. We remark that the complexity mainly depends on the ring $\mathcal{D}_H$ which can be much smaller than the ciphertext space $R_q^D \subseteq \mathbb{Z}_q[X]^D$ (see Section 4.3). It makes our VC scheme show better efficiency (in both asymptotic and concrete cost) than a naive VC (over the ciphertext space) without our homomorphic hash functions (see Section 4.4 for detailed analysis).

Applications to VC for SIMD Computations. Besides the application to HE computations, the use of ε-universal family of homomorphic hash functions

can have broader applications in improving prover efficiency in VC over polynomial rings, i.e., when proving and verifying the computations over a polynomial ring. By combining this observation with the "packing" techniques of HE [SV14] one can obtain a VC scheme for SIMD (single-instruction multiple-data) operations where the prover's costs are less dependent on the number of (parallel) inputs. A bit more in detail, with the packing techniques of [SV14] one can encode a vector $(v_i)_i$ of m elements of $\mathbb{Z}_t$ into a polynomial $p \in R_t$ (such that $d_f \geq m$) in such a way that the result of computing $\hat{g}((p_j)_j)$ over $R_t := \mathbb{Z}_t[X]/(f)$ can be decoded to the vector $(g((v_{j,i})_j))_i$ over $\mathbb{Z}_t$. By using a homomorphic hash from R_t to $\mathbb{Z}_t[X]/(h)$ we can obtain a prover time which depends on $|g| \cdot d_h + d_f$, as opposed to $|\hat{g}| \approx |g| \cdot d_f$.[6] We remark that $d_h \approx \log_t \lambda$ (when t is prime) while d_f can be as large as the number of parallel inputs.

4 Instantiating Our VC Scheme

In this section, we provide concrete instantiations of the building blocks for our generic scheme presented in the previous section. In particular, our novel contributions are the constructions of two homomorphic hash functions. We also give a detailed efficiency analysis with example parameters.

4.1 SHE - The BV Homomorphic Encryption Scheme

As an instantiation of SHE, we exploit the BV scheme [BV11] which allows homomorphic evaluation of circuits of limited multiplicative depth. The advantage of the BV scheme for our purpose is that its homomorphic additions and multiplications over ciphertexts are composed of arithmetic operations over R_q^D only.[7] As a result, this scheme can be easily combined with the homomorphic hash functions (which will be described in the following subsection) defined in our generic VC scheme.

Parameters. Let q and t ($t < q$) be coprime integers, $f \in \mathbb{Z}[X]$ be a monic polynomial of degree d_f, and $R := \mathbb{Z}[X]/(f)$. The plaintext space is $R_t := \mathbb{Z}_t[X]/(f)$, and the ciphertext space is $R_q^D = (\mathbb{Z}_q[X]/(f))^D$ where $D(\geq 2)$ bounds the total degree of a multi-variate polynomial which can be evaluated on ciphertexts, i.e., products of at most $D-2$ input ciphertexts are allowed.

Homomorphic Operations. A ciphertext $c = (c_0, c_1, \ldots, c_{D-1}) \in R_q^D$ is identified as a polynomial $c(Y) \in R_q[Y]$ of degree at most $D-1$ as follows:

$$c(Y) = \sum_{i=0}^{D-1} c_i Y^i.$$

[6] We assume that the cost of basic operation over a ring ($\mathbb{Z}_t[X]/(h)$ or $\mathbb{Z}_t[X]/(f)$) depends on its degree (d_h or d_f) for simplicity.

[7] This is not the case in other schemes, e.g., BGV [BGV12] or FV [FV12] schemes where multiplication of ciphertexts accompanies rounding or bitwise operations.

Then, addition and multiplication of two ciphertexts $c = \mathsf{Enc}(m), c' = \mathsf{Enc}(m')$ are defined, respectively, by the usual addition and multiplication in $R_q[Y]$:

- $c_{\mathrm{add}}(Y) := c(Y) + c'(Y)$, i.e., $c_{\mathrm{add}} := (c_0 + c'_0, c_1 + c'_1, \ldots, c_{D-1} + c'_{D-1})$.
- $c_{\mathrm{mult}}(Y) := c(Y) \cdot c'(Y)$, i.e., $c_{\mathrm{mult}} := (\hat{c}_0, \hat{c}_1, \ldots, \hat{c}_{D-1})$ where $\sum_{i \geq 0}^{D-1} \hat{c}_i Y^i = c(Y) \cdot c'(Y)$.

The *correctness* ($\mathsf{Dec}(c_{\mathrm{add}}) = m + m'$, $\mathsf{Dec}(c_{\mathrm{mult}}) = m \cdot m'$) is guaranteed only if the degree (in Y) of result ciphertext (c_{add} or c_{mult}) does not exceed $D - 1$. We remark that a fresh ciphertext is of degree 1 (in Y), i.e., $c_i = 0$ for all $i > 2$, and the correctness is guaranteed for the computation represented by a (multivariate) polynomial of total degree at most $D - 1$.

We refer to the full version of this article for the description of other algorithms ($\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$) of BV scheme and the concrete conditions for the *correctness* and *security* of BV scheme.

4.2 Argument System - The GKR Protocol over Rings

The GKR protocol [GKR08] is a (public-coin) interactive proof system for arithmetic circuits over a finite field. In [CCKP19], Chen et al. showed that the protocol can be generalized to handle arithmetic circuits over a finite ring. We exploit this GKR protocol over rings [CCKP19] (with Fiat-Shamir) as our instantiation of argument system, since it can efficiently prove and verify arithmetic of rings[8] which constitutes the range $\mathcal{D}_H$ ($\mathbb{Z}_q[X]/(h)$ or $(\mathbb{Z}_q[X]/(h))^D$) of our hash functions (Section 4.3).

One drawback of the GKR protocol is that the circuit (to be verified) should be log-space uniform and be layered in low depth for efficient verification. A line of work has shown that many computations of interest are in this form [CMT12, Tha13] or can be converted to this form [WTs+18]. Therefore, we (plausibly) assume that the given circuit is log-space uniform and that a succinct description of the circuit can be found during the preprocessing phase (e.g., $\mathsf{KeyGen}_{PK}(g)$ finds such description for g and puts it into PK_g). We remark that, in our instantiation, the circuit already has low depth to be supported by the BV scheme.

In this section, we recall the definition of Galois rings, the Schwartz-Zippel lemma for rings, and give a summary of the GKR protocol over rings. In the full version of this article we add a detailed description of the GKR protocol [GKR08, Tha13] and its generalization to rings [CCKP19]. In this section, rings are commutative rings with multiplicative identity 1.

Galois Rings. Galois rings play a central role in the GKR protocol over rings and in our instantiation of hash functions. Galois rings are a natural generalization of Galois fields $\mathsf{GF}(p^n) = \mathbb{Z}_p[X]/(f)$, and proofs of the following properties of Galois rings can be found in [Wan03].

[8] Usual argument systems deal mainly with arithmetic of a field, and it requires to represent arithmetic of a ring by that of a field, resulting in substantial inefficiency.

Definition 3 (Galois ring). *A Galois ring is a ring of the form*

$$\mathbb{Z}_{p^e}[X]/(f)$$

where p is a prime number, e is a positive number, $f \in \mathbb{Z}_{p^e}[X]$ is a monic polynomial whose reduction modulo p is an irreducible polynomial in $\mathbb{Z}_p[X]$.

We remark that a Galois ring $\mathbb{Z}_{p^e}[X]/(f)$ has many more invertible elements than the base ring $\mathbb{Z}_{p^e}$:

Lemma 2 (Units of Galois ring). *In a Galois ring $R_q := \mathbb{Z}_{p^e}[X]/(f)$, the set of units of R_q is $R_q \setminus pR_q$, i.e., the elements which are not divisible by p. In fact, we have a ring isomorphism $R_q/pR_q \cong \mathbb{F}_{p^{d_f}}$ where d_f is the degree of f.*

Schwartz-Zippel Lemma for Rings. We now present the Schwartz-Zippel lemma for rings, specifically, we focus on the case of Galois rings which is closely related to our instantiation.

Definition 4 (Sampling set). *Let R be a finite ring and $A \subset R$. We call A a sampling set if*

$$\text{for all } x, y \in A \text{ such that } x \neq y, \quad x - y \text{ is invertible.}$$

Lemma 3 (Schwartz-Zippel). *Let R be a finite ring, and let $A \subset R$ be a sampling set. Let $f : R^n \to R$ be an n-variate nonzero polynomial of total degree D. Then*

$$\Pr_{x \leftarrow A^n}[f(x) = 0] \leq \frac{D}{|A|}.$$

Examples of Sampling Set. In a ring $\mathbb{Z}_{p^e}$ with p prime, $A = \{0, 1, .., p-1\}$ is a maximal sampling set with cardinality p. In a Galois ring $\mathbb{Z}_{p^e}[X]/(f)$ where f is a monic polynomial of degree d_f, the set $\{a_0 + a_1 X + \cdots + a_{d_f-1} X^{d_f-1} \mid a_i \in A\}$ is a maximal sampling set with cardinality p^{d_f}.

In the following, we borrow the result of [CCKP19], the generalized GKR protocol on the circuit over Galois rings. The soundness of the protocol is guaranteed by the Schwartz-Zippel lemma (Lemma 3), thus it depends on the size of sampling set, e.g., p^{d_f} in the following.

Theorem 2 (GKR protocol over Galois rings [CCKP19]). *Let $R_q := \mathbb{Z}_{p^e}[X]/(f)$ be a Galois ring where f is of degree d_f. Let $C : R_q^n \to R_q$ be an arithmetic circuit over R_q taking n inputs and outputting 1 output. Let C be of size (the number of arithmetic gates contained) S and depth d. Then, there exists an interactive protocol (with public coins) for C with perfect completeness and soundness $\frac{7d\log S}{p^{d_f}}$, which requires the same number of operations (over R_q) for a prover and a verifier as the GKR protocol over a finite field (where the required operations are over a finite field).*

Efficiency of GKR protocol. The latest refinement [XZZ+19] of the GKR protocol reduced the prover's cost to $O(S)$. Since their technique can also be adapted to the protocol over Galois rings (their technique only uses addition, multiplication, and bookkeeping which are all available in arbitrary rings), the complexity of our instantiation is also $(T_{\mathcal{P}}^{\Pi}, T_{\mathcal{V}}^{\Pi}, C^{\Pi})$[9] $= (O(S),\ O(n+d\log S)$[10]$,\ O(d\log S))$. We remark that the space complexity of a verifier $\mathcal{V}$ can be $O(\log S)$ only (without increasing other cost) and that the time complexity $O(d\log S)$ of $\mathcal{V}$ can be regarded as $o(S)$ since d is a small constant in the usual utilization of BV scheme.

4.3 Our Homomorphic Hash Functions Realizations

In this section, we present explicit constructions of two families of ε-universal homomorphic hash functions on some domain $\mathcal{D} \subset \mathbb{Z}_q[X]^D$ with $q = p^e$ for a prime p. Both of our hash function families, taking as input D polynomials of $\mathbb{Z}_q[X]$, are based on the map of modulo reduction by a polynomial $h \in \mathbb{Z}_q[X]$, which is a natural generalization of the evaluation map $(f \in \mathbb{Z}_q[X] \to f(r) \in \mathbb{Z}_q)$ exploited in the previous works [FGP14, FNP20]. The range of our hash function families are $(\mathbb{Z}_q[X]/(h))^D$ or $\mathbb{Z}_q[X]/(h)$ where $h \in \mathbb{Z}_q[X]$ is a monic polynomial whose reduction modulo p is irreducible in $\mathbb{Z}_p[X]$, i.e., $\mathbb{Z}_q[X]/(h)$ is a Galois ring.

4.3.1 Homomorphic Hash Function - I. Single Hash

We first give the definition of our hash functions specifying the domain $\mathcal{D}$.

Definition 5 (Single Hash Function). *Let N, D be positive integers and $q = p^e$ for a prime p, and let $\mathcal{D} = \{(z_i)_{i=0}^{D-1} \in \mathbb{Z}_q[X]^D : deg_X(z_i) \le N\}$. For a monic polynomial $h \in \mathbb{Z}_q[X]$, the hash function H_h on $\mathcal{D}$ is defined as follows.*

$$\begin{aligned} H_h : \mathcal{D} \subset \mathbb{Z}_q[X]^D &\longrightarrow (\mathbb{Z}_q[X]/(h))^D \\ (z_i)_{i=0}^{D-1} &\mapsto (z_i \pmod h)_{i=0}^{D-1} \end{aligned}$$

where $z_i \pmod h$ is the remainder of z_i when divided by h in $\mathbb{Z}_q[X]$.

We can gather these hash functions into a family of hash functions which satisfies the ϵ-universality as follows.

Theorem 3. *Let $N, D, d_{\mathcal{H}}$ be positive integers and $q = p^e$ for a prime p. On $\mathcal{D} = \{(z_i)_{i=0}^{D-1} \in \mathbb{Z}_q[X]^D : deg_X(z_i) \le N\}$, the family of functions $\mathcal{H} := \{H_h : h \in \mathbb{Z}_q[X]$ is monic, degree-$d_{\mathcal{H}}$, and irreducible (in $\mathbb{Z}_p[x]$)$\}$ is homomorphic and ε-universal for $\varepsilon = \frac{2N}{p^{d_{\mathcal{H}}}}$.*

In other words, for all $z, z' \in \mathcal{D}$ such that $z \neq z'$,

$$\Pr[H(z) = H(z') : H \xleftarrow{\$} \mathcal{H}] \le \frac{2N}{p^{d_{\mathcal{H}}}}.$$

[9] We refer to Lemma 1 for this notation.

[10] Here, we assume that the wiring predicate [CMT12] of the circuit is computable in $O(\log S)$ complexity. Generally, if the circuit is log-space uniform, the cost of verifier has an additional $O(poly(\log S))$ term.

Proof. The homomorphic property of hash functions $H_h \in \mathcal{H}$ follows from that of the modulo reduction by $h \in \mathbb{Z}_q[X]$. For the probability of a collision, let $\Delta \in \mathbb{Z}_q[X]$ be a non zero element among the components of $z - z'$. Then, Δ is a non zero polynomial of degree at most N, and $H_h(z) = H_h(z')$ implies that Δ has $h \in \mathbb{Z}_q[X]$ as a factor, which is equivalent to that Δ has h as a factor in $\mathbb{Z}_p[X]$ (after modulo reduction by p).[11] Therefore,

$$\begin{aligned}\Pr[H(z) = H(z') : H \xleftarrow{\$} \mathcal{H}] &\leq \Pr[h \text{ divides } \Delta \text{ in } \mathbb{Z}_p[X] : h \xleftarrow{\$} A(d_{\mathcal{H}}, p)] \\ &\leq \frac{N}{d_{\mathcal{H}}} \times \frac{1}{I(d_{\mathcal{H}}, p)} \quad \leq \frac{2N}{p^{d_{\mathcal{H}}}},\end{aligned}$$

where $A(n,p)$ (and $I(n,p)$) denote the set (resp., the number) of monic irreducible polynomials of degree n in $\mathbb{Z}_p[X]$; the second inequality follows from the fact that, in $\mathbb{Z}_p[X]$, a degree-N polynomial can have at most N/d irreducible factors of degree d; the third inequality follows from the lower bound of $I(n,p)$ in the following lemma. □

Lemma 4. *Let μ be the Möbius function,[12] p be a prime number, and $n \geq 1$ be an integer. Let $I(n,p)$ be the number of monic irreducible polynomials in $\mathbb{Z}_p[X]$ of degree n. Then,*

$$I(n,p) = \frac{1}{n} \sum_{d|n} \mu\left(\frac{n}{d}\right) p^d.$$

It also holds for all n that $I(n,p) \geq \frac{1}{n}(p^n - 2p^{\lfloor \frac{n}{2} \rfloor})$ and $I(n,p) \geq \frac{p^n}{2n}$. Moreover, if n is also a prime number, then $I(n,p) = \frac{1}{n}(p^n - p)$. Finally, asymptotically we have $I(n,p) \underset{n\to\infty}{\sim} \frac{p^n}{n}$.

Proof. (sketch) We refer to [VZGG13, Lemma 14.38] for details. Let $A(d,p)$ denote the set of monic irreducible polynomials of degree d in $\mathbb{Z}_p$. Then, from the fact that $X^{p^n} - X = \prod_{d|n} \prod_{\phi \in A(d,p)} \phi$, computing the degree of that product and using the Möbius inversion formula, we get the equation on $I(n,p)$. If n is prime, then this formula has only two summands, namely for $d = 1$, and $d = n$. From the definition of μ, we get $I(n,p) = \frac{1}{n}(p^n - p)$. The bounds $\frac{1}{n}p^n \geq I(n,p) \geq \frac{1}{n}(p^n - 2p^{\lfloor \frac{n}{2} \rfloor})$ are not difficult to prove for all n. This implies the asymptotic statement. For $p^n \geq 16$, the lower bound above directly implies $I(n,p) \geq \frac{p^n}{2n}$. For $n = 1$ clearly $I(1,p) = p \geq \frac{p}{2n}$. Finally for the remaining cases ($p^n = 4, 8, 9$) it can be checked that $I(n,p) \geq \frac{p^n}{2n}$ also holds. □

Remark 4 (Setting ε). We can set ε of the homomorphic hash family negligibly small, e.g., $d_{\mathcal{H}} \approx \lambda \log_p 2$ gives that $\varepsilon \approx \frac{1}{2^\lambda}$. In case of our instantiation with BV,

[11] More precisely, the argument follows if Δ is not zero when reduced modulo p. Otherwise, $\Delta = p^k \delta$ for some $k < e$ and a polynomial δ which is not zero when reduced modulo p, and δ has h as a factor in $\mathbb{Z}_p[X]$: $h|\Delta$ gives that, by division, $\delta(X) = h(X)Q(X) + p^{e-k}r(X)$ in $\mathbb{Z}_q[X]$ and h is a factor of δ in $\mathbb{Z}_p[X]$.

[12] For $n \in \mathbb{Z}^+$, the function is defined as follows: if n is square-free with k prime factors, $\mu(n) = (-1)^k$; if $n = 1, \mu(n) = 1$; otherwise, $\mu(n) = 0$.

the degree N of polynomials in $\mathcal{D}$ is bounded by $(d_f - 1)(D - 1)$, and ε can be bounded by $\frac{2(d_f-1)(D-1)}{p^{d_\mathcal{H}}}$.

Remark 5 (Sampling h). For an efficient instantiation of above homomorphic hash functions, one has to efficiently sample (uniformly randomly) an h from the set $A(n, p)$ of monic irreducible polynomials in $\mathbb{Z}_p[X]$ of degree n. We explain how to do so in Section 4.3.3.

4.3.2 Homomorphic Hash Function - II. Double Hash

Recall that ciphertext additions and multiplications of BV scheme (Section 4.1) respectively correspond to the additions and multiplications of polynomials in $R_q[Y]$ and that, in our generic VC scheme (Section 3.2), those ciphertext operations are carried on $(\mathbb{Z}_q[X])[Y]$ (polynomials in Y having coefficients from $\mathbb{Z}_q[X]$) by postponing the modulo f operation. Then, we can define a homomorphic hash with much smaller range $\mathbb{Z}_q[X]/(h)$, instead of $(\mathbb{Z}_q[X]/(h))^D$ in the previous section.

Definition 6 (Double Hash Function). *Let N and D be positive integers, and let $\mathcal{D} = \{z \in \mathbb{Z}_q[X][Y] : deg_X(z) \leq N \text{ and } deg_Y(z) < D\}$. For a monic polynomial $h \in \mathbb{Z}_q[X]$ and an element $r \in \mathbb{Z}_q[X]/(h)$, the hash function $H_{r,h}$ on $\mathcal{D}$ is defined as follows.*

$$\begin{array}{ccccc} H_{r,h} : \mathcal{D} \subset \mathbb{Z}_q[X][Y] & \longrightarrow & (\mathbb{Z}_q[X]/(h))[Y] & \longrightarrow & \mathbb{Z}_q[X]/(h) \\ \sum_{i=1}^{D-1} z_i Y^i & \mapsto & \sum_{i=1}^{D-1} \bar{z}_i Y^i & \mapsto & \sum_{i=1}^{D-1} \bar{z}_i r^i \end{array}$$

where $\bar{z}_i := z_i \pmod h$ is the remainder of z_i when divided by h in $\mathbb{Z}_q[X]$.

Note that $H_{r,h}$ is indeed the composition of H_h (Definition 5) and an evaluation map $(z(Y) \to z(r))$ on $(\mathbb{Z}_q[X])[Y]$. Similarly as the case of single hash functions (Section 4.3.1), we can gather this hash functions into a family of hash functions which satisfies the ϵ-universality as follows.

Theorem 4. *Let $N, D, d_\mathcal{H}$ be positive integers and $q = p^e$ for a prime p. On $\mathcal{D} = \{z \in \mathbb{Z}_q[X][Y] : deg_X(z) \leq N \text{ and } deg_Y(z) < D\}$, the family of functions $\mathcal{H} := \{H_{r,h} : h \in \mathbb{Z}_q[X]$ is monic, degree-$d_\mathcal{H}$, and irreducible (in $\mathbb{Z}_p[x]$), and $r \in \mathbb{Z}_q[X]/(h)$ is from the maximal* sampling set (Definition 4) *of $\mathbb{Z}_q[X]/(h)\}$ is homomorphic and ε-universal for $\varepsilon = \frac{2N+D-1}{p^{d_\mathcal{H}}}$.*

In other words, for all $z, z' \in \mathcal{D}$ such that $z \neq z'$,

$$\Pr[H(z) = H(z') : H \xleftarrow{\$} \mathcal{H}] \leq \frac{2N + D - 1}{p^{d_\mathcal{H}}}.$$

Proof. As we noted, $H_{r,h}$ is the composition of H_h (Definition 5) and an evaluation map $(z(Y) \to z(r))$ on $R'_q[Y]$ where $R'_q := \mathbb{Z}_q[X]/(h)$. Therefore, the homomorphic property of $H_{r,h} \in \mathcal{H}$ follows from that of the H_h (Theorem 3)

and that of the evaluation map ($z(Y) \in R'_q[Y] \to z(r) \in R'_q$). For the probability of a collision, let $\Delta := z - z' \in \mathbb{Z}_q[X][Y]$. Then, Δ is a non zero polynomial of degree at most N in X and of degree less than D in Y. In the following, let $A = A(d_{\mathcal{H}}, p)$ be the set of monic irreducible polynomials of degree n in $\mathbb{Z}_p[X]$, and let B be the maximal sampling set (Definition 4) of R'_q. Then,

$$\begin{aligned}\Pr[H(z) = H(z') : H \xleftarrow{\$} \mathcal{H}] &\leq \Pr_{h \leftarrow A}[H_h(\Delta) = 0 \in R'_q[Y]] \\ &\quad + \Pr_{r \leftarrow B}[H_h(\Delta)(r) = 0 | H_h(\Delta) \neq 0 \in R'_q[Y]] \\ &\leq \frac{2N}{p^{d_{\mathcal{H}}}} + \frac{D-1}{p^{d_{\mathcal{H}}}},\end{aligned}$$

where the second inequality follows from Theorem 3 and Lemma 3: on the right side, the first summand is the result of Theorem 3 while the second summand follows from the fact that the degree of $H_h(\Delta)$ in Y is less than D and that the size of the maximal sampling set B of R'_q is $p^{d_{\mathcal{H}}}$ since R'_q is a Galois ring (see examples following Lemma 3). □

We can also set ε of the double hash family negligibly small: in our instantiation with BV, since the degree N of polynomials in $\mathcal{D}$ is bounded by $(d_f - 1)(D - 1)$,

$$\varepsilon \leq \frac{2(d_f - 1)(D-1) + D - 1}{p^{d_{\mathcal{H}}}}. \tag{1}$$

Remark 6 (Comparison to Single Hash). Utilizing double hashes instead of single hashes gives better efficiency. With double hash, each addition (resp. multiplication) gate on the ciphertext space $(\mathbb{Z}_q[X])^D = \mathbb{Z}_q[X][Y]$ maps to each addition (resp. multiplication) gate of $R'_q = \mathbb{Z}_q[X]/(h)$ while the single hash maps each of them to at most D additions (resp. D^2 multiplications and D additions) of R'_q. See Section 4.4.2 for detailed analysis.

Remark 7 (Sampling r). Recall that the Galois ring $\mathbb{Z}_q[X]/(h)$ can be identified as a set $\{\sum_{i=0}^{d_h - 1} a_i X^i : a_i \in \{0, 1, 2, \ldots, q-1\}\}$ where d_h is the degree of h. Therefore, sampling (uniformly randomly) r from the maximal *sampling set* (Definition 4) of $\mathbb{Z}_q[X]/(h)$ is simple, since the set can be identified as a set $\{\sum_{i=0}^{d_h - 1} a_i X^i : a_i \in \{0, 1, 2, \ldots, p-1\}\} \subseteq \mathbb{Z}_q[X]/(h)$.

4.3.3 Efficient, Public-Coin Sampling of h

We now turn our attention to an efficient sampling method of a monic irreducible polynomial h of degree d_h in $\mathbb{Z}_p[X]$ for our hash functions (Section 4.3). We first recall a method that is based on a textbook algorithm [Ben81, VZGG13]. It samples an irreducible polynomial by repeatedly sampling a random monic polynomial, and then checking if it is irreducible by verifying whether it is coprime with $X^{p^i} - X$ for every $i \leq d_h/2$. In our version, we slightly change this algorithm to make explicit the number of public random coins needed to make it fail with negligible probability.

Theorem 5 (Ben-Or's Generation of Irreducible Polynomials [Ben81]). *There exists an algorithm that, on input $2d_h(d_h-1)\lambda$ random elements of $\mathbb{Z}_p$, returns a uniformly sampled monic irreducible polynomial of degree d_h in $\mathbb{Z}_p[X]$, takes expected number of $\tilde{O}(d_h^2 \log p)$ operations in $\mathbb{Z}_p$ ($\tilde{O}$ hides logarithmic factors in d_h), and fails with probability $\leq 2^{-\lambda}$.*

In the full version we include a description of the algorithm and the proof.

A drawback of the above algorithm is its rejection sampling nature, which in a public coin instantiation (especially when applying the Fiat-Shamir heuristic to our protocol) forces us to sample a significant number of random coins ($2d_h(d_h-1)\lambda$ $\mathbb{Z}_p$-elements) to make the probability of failure negligible.

To avoid this, we propose an alternative sampling method that achieves a similar complexity and uses much less random coins, $2d_h$ $\mathbb{Z}_p$-elements. It is based on the following observation: Let $\mathbb{F}_{p^{d_h}} := \mathbb{Z}_p[X]/(\phi)$ be a finite field. Then, it suffices to sample an element of $\mathbb{F}_{p^{d_h}}$ which is not contained in any of the subfields $\mathbb{F}_{p^k}$ with $k|d_h$, since the minimal polynomial of such element is monic, irreducible, and of degree d_h in $\mathbb{Z}_p[X]$. It turns out that given a generator α of the multiplicative group $\mathbb{F}^{\times}_{p^{d_h}}$, these sampleable elements are exactly α^j where $j = \sum_{i=0}^{d_h-1} a_i p^i$ and $(a_0, a_1, \ldots, a_{d_h-1}) \notin \bigcup_{k|d_h, k\neq d_h} \mathtt{Bad}_k$, where[13]

$$\mathtt{Bad}_k = \{(\vec{v}^{\,d_h/k}) \in \mathbb{Z}_p^{d_h} : \vec{v} \in \{0, \ldots, p-1\}^k\}.$$

Theorem 6 (Sampling Irreducible Polynomials). *The following algorithm, on input $2d_h$ random elements of $\mathbb{Z}_p$, returns a uniformly sampled monic irreducible polynomial of degree d_h in $\mathbb{Z}_p[X]$, takes $O(d_h M(d_h) \log p)$ operations in $\mathbb{Z}_p$ where $M(d_h)$ denotes the complexity of multiplying polynomials of degree d_h in $\mathbb{Z}_p[X]$, and fails with probability $\leq 4p^{-(d_h-1)} \approx 2^{-\lambda}$.*

[Algorithm]

- Input: A prime p and a degree $d_h \in \mathbb{Z}_{>0}$.
- Random coins: $\rho_0^{(1)}, \ldots, \rho_{d_h-1}^{(1)}, \rho_0^{(2)}, \ldots, \rho_{d_h-1}^{(2)} \in \mathbb{Z}_p$.
- Output: A monic irreducible polynomial of degree d_h in $\mathbb{Z}_p[X]$.
- Setup:
  - Fix a finite field $\mathbb{F}_{p^{d_h}} := \mathbb{Z}_p[X]/(\phi)$ with an irreducible polynomial $\phi \in \mathbb{Z}_p[X]$ of degree d_h.
  - Let α be a generator of the multiplicative group $\mathbb{F}^{\times}_{p^{d_h}}$, compute and store $\alpha_i = \alpha^{p^i}$ for $i \in \{1, \ldots, d_h - 1\}$.
- Procedure:
  1. If $(\rho_0^{(1)}, \ldots, \rho_{d_h-1}^{(1)}) \notin \bigcup_{k|d_h, k\neq d_h} \mathtt{Bad}_k$, set $\beta = \prod_{i=0}^{d_h-1} \alpha_i^{\rho_i^{(1)}}$, go to 4.
  2. Else if $(\rho_0^{(2)}, \ldots, \rho_{d_h-1}^{(2)}) \notin \bigcup_{k|d_h, k\neq d_h} \mathtt{Bad}_k$, set $\beta = \prod_{i=0}^{d_h-1} \alpha_i^{\rho_i^{(2)}}$, go to 4.
  3. Else, return `fail`.
  4. Find the minimal polynomial h of β, using the algorithm in [Sho99].
  5. Return h

[13] $\vec{v}^{\,\ell}$ denotes ℓ concatenations of $\vec{v}$.

See the full version of the paper for a proof of this result.

Remark 8. Using Fast Fourier Transform algorithms or the Schönhage-Strassen algorithm we can set $M(d_h) = \tilde{O}(d_h)$ and the complexity above becomes $\tilde{O}(d_h^2 \log p)$ (where $\tilde{O}$ hides logarithmic factors in d_h).

Remark 9. If d_h is prime, we only need the coins $\rho_i^{(1)}$ as step 1 succeeds with overwhelming probability since $|\bigcup_{k|d_h, k\neq d_h} \mathtt{Bad}_k| = |\mathtt{Bad}_1| = p \ll p^{d_h}$.

4.4 Efficiency Analysis

In this section, we analyze the efficiency of the generic VC scheme (Section 3.2) instantiated with the concrete building blocks described so far. As before, let $R_t := \mathbb{Z}_t[X]/(f)$, let $g : (R_t)^n \to R_t$ denote the (delegated) computation of degree less than D over the plaintext space R_t, and $\mathcal{P}$ computes (then proves) the function $\hat{g} : (R_q^D)^n \to \mathbb{Z}_q[X]^D$ (not R_q^D) that describes $\mathsf{SHE.Eval}_{\mathsf{evk}}(g, \cdot)$ without reduction modulo f.

4.4.1 Combining Instantiations

The aggregation of building blocks can be summarized as follows:

SHE - BV scheme: In the setup, take a polynomial f of (large enough) degree d_f and a ciphertext modulus q satisfying 2^λ security and correctness of computation of any multivariate polynomial of degree $< D$ for plaintext modulus t, which sets the ciphertext space $R_q^D = (\mathbb{Z}_q[X]/(f))^D$. See more details in the full version.

Homomorphic Hash: Note that the ciphertext space R_q^D can be identified as (a subset of) $\mathbb{Z}_q[X]^D$ or $\mathbb{Z}_q[X][Y]$.[14] We can use our single hash or double hash, whose domain and range are as follows.

* Single: $\{(z_i)_{i=0}^{D-1} \in \mathbb{Z}_q[X]^D : deg_X(z_i) \leq N\} \longrightarrow (\mathbb{Z}_q[X]/(h))^D$
* Double: $\{z \in \mathbb{Z}_q[X][Y] : deg_X(z) \leq N \text{ and } deg_Y(z) < D\} \longrightarrow \mathbb{Z}_q[X]/(h)$

In both cases, $N = (d_f - 1)(D - 1)$ as noted before (in Remark following Theorem 3, 4). Note, in the range of both hash functions, that $\mathbb{Z}_q[X]/(h)$ is a Galois ring.

Argument System - GKR protocol over $\mathbb{Z}_q[X]/(h)$: Since the image of computations under the hash functions are in the ring $\mathbb{Z}_q[X]/(h)$ which is a Galois ring, we can use the GKR protocol to prove and verify such computations.

Security: We remark that the polynomial $h \in \mathbb{Z}_q[X]$ (where $q = p^e$) should be chosen of degree $\log_p \lambda$ to guarantee $\approx \frac{1}{2^\lambda}$ universality (and $\approx \frac{1}{2^\lambda}$ soundness) in the hash functions (resp. in the GKR protocol), from which our scheme gets $\approx \frac{1}{2^\lambda}$ soundness (see Theorem 1, 2, 3, 4).

[14] For this, we skip the modulo reduction by f at the (delegated) computation.

4.4.2 Complexity of the Instantiation

As expected from the complexity analysis (Lemma 1) in Section 3.2, the efficiency of the instantiation mainly depends on the range space (ring) of hash functions and the costs of the argument system on that space.

Theorem 7. *In the instantiation of our VC scheme with soundness $2^{-\lambda}$ and the* BV *scheme having $R_q := \mathbb{Z}_q[X]/(f)$ as a ciphertext ring, assume that a verifier delegates a function $g : R_t^n \to R_t$ of degree less than D, which is described by an arithmetic circuit C over R_t of size S and depth d. Then, the required complexity $(T_{\mathcal{P}}, T_{\mathcal{V}}, C_{\mathcal{C}})$*[15] *measured by the number of operations (or elements) of $\mathbb{Z}_q$ is as follows.*

(i) with Single hash:

$$(\tilde{O}((n+D^2)d_f+\lambda D^2 S),\ \tilde{O}((n+D^2)d_f+\lambda d\log(D^2S)),\ O(D^2d_f+\lambda d\log(D^2S)))$$

(ii) with Double hash:

$$(\tilde{O}((n+D^2)d_f+\lambda S),\quad \tilde{O}((n+D^2)d_f+\lambda d\log S),\quad O(D^2d_f+\lambda d\log S))$$

where d_f is the degree of $f \in \mathbb{Z}_q[X]$ and $\tilde{O}$ hides the logarithmic factors. In $T_{\mathcal{P}}$, we assume that the prover $\mathcal{P}$ already has the result of computation $\hat{g} : (R_q^D)^n \to \mathbb{Z}_q[X]^D$ over the ciphertexts.

Proof. (sketch) It follows from the fact that the homomorphic image of computation $\hat{g} : (R_q^D)^n \to \mathbb{Z}_q[X]^D$ under the single or double hash function, respectively, is composed of $O(D^2S)$ operations or $O(S)$ operations over $\mathbb{Z}_q[X]/(h)$ (with $\deg h = O(\lambda)$), respectively. The output $\hat{g}(c_1, \ldots, c_n) \in \mathbb{Z}_q[X]^D \subset \mathbb{Z}_q[X][Y]$ is a polynomial of degree $O(D)$ in Y and of $O(Dd_f)$ in X, hence is composed of $O(D^2d_f)$ elements of $\mathbb{Z}_q$, and the cost of evaluating hash function (on n input and 1 output) is $\tilde{O}((n+D^2)d_f)$. See full version for the full proof. □

Note that computing $\hat{g} : (R_q^D)^n \to \mathbb{Z}_q[X]^D$ over ciphertexts costs $\Omega(d_f D^2 S)$ operations over $\mathbb{Z}_q$, and is roughly $\times D$ costly than the original ciphertext computation $\mathsf{SHE.Eval}_{\mathsf{evk}}(g, \cdot) : (R_q^D)^n \to R_q^D$ (with mod f) which costs $\Omega(d_f DS)$. However, this gap is not significant given that the degree D of computation is not large.

Remark 10 (Ciphertext Computation vs Proof Generation). Since $d_f = \Omega(\sqrt{\lambda}\log q)$ for the security of BV scheme (from the hardness of LWE), double hash makes the proof generation cost $T_{\mathcal{P}} = \tilde{O}((n+D^2)d_f + \lambda S)$ asymptotically *negligible* to the cost of computing $\hat{g}$ (while single hash does not). In other words, with our scheme, there is no significant additional overhead for $\mathcal{P}$ to prove the correctness of its computation, as will be also demonstrated with concrete example parameters in the following section.

[15] Each signifies the time complexity of $\mathcal{P}$, that of $\mathcal{V}$, and the communication cost.

Circuit			BV scheme				GKR & Hash	Security	
g	n	d	D	$\log t$	$\log q$	$\log d_f$	d_h	λ_{BV}	λ_s
Inn Prod.	2^8	2	3	8	54	11	136		128.4
Inn Prod.	2^8	2	3	16	73	12	136	≥ 128	128.4
Inn Prod.	2^{10}	2	3	16	73	12	136		128.2
Poly Eval.	2^{10}	2	4	2	62	11	136	117.5	128.1
Poly Eval.	2^{10}	4	6	2	110	12	137		128.8
Poly Eval.	2^{10}	4	6	16	187	13	137	≥ 128	128.7
Poly Eval.	2^{12}	16	18	16	685	15	138		128.6

Table 2. Example parameters for our VC scheme: λ_{BV} and λ_s respectively denote the bit security of BV scheme and our VC scheme.

4.4.3 Concrete Parameters and Examples

To demonstrate the efficiency of our VC, we give some explicit parameters for the example computation. For $R_t := \mathbb{Z}_t[X]/(f)$, assume that a verifier delegates a (multivariate) polynomial $g : R_t^u \to R_t$ of (total) degree D. We let g be one of the two examples given below:

- Inner Product of two input vectors of dimension n over R_t, i.e.,

$$\begin{array}{rccc} g: & R_t^n \times R_t^n & \longrightarrow & R_t \\ & ((a_1, a_2, \ldots, a_n), (b_1, b_2, \ldots, b_n)) & \mapsto & \sum_{i=1}^n a_i b_i. \end{array}$$

- (Degree-d_F) Polynomial Evaluation on n parallel input points of R_t, i.e.,

$$\begin{array}{rccc} g: & R_t^n \times R_t^{d_F+1} & \longrightarrow & R_t^n \\ & ((a_1, a_2, \ldots, a_n), (F_0, F_1, \ldots, F_{d_F})) & \mapsto & (\sum_{i=0}^{d_F} F_i a_1^i, \sum_{i=0}^{d_F} F_i a_2^i, \ldots, \sum_{i=0}^{d_F} F_i a_n^i). \end{array}$$

We refer to the full version for the detailed description of above computations and the derivation of parameters.

In Table 2, we present several parameters for our VC scheme with double hash for those computations varying n, the depth d (equivalent to the degree of the input polynomial in the second example) of the circuit $g : R_t^u \to R_t$, and the plaintext modulus t.

Efficiency Improvement. With Table 2, we can give more concrete analysis on the efficiency of our VC scheme. In a naive approach without our hashing, the prover should generate a proof on the computation over $R_q = \mathbb{Z}_q[X]/(f)$ where the degree d_f of f is 2^{11}–2^{15} in above examples. In contrast, in our scheme with hashing, the proof is generated on the computation over $\mathbb{Z}_q[X]/(h)$ where the degree d_h of h is only 136–139 ($\approx \lambda_s$ as expected). Therefore, we can expect roughly $\times \frac{d_f}{d_h} \approx \times 15$–$235$ improvement in the cost of proof generation and the size of proof. It also implies that the cost of proof generation is not significant compared to the ciphertext computation, since the former is done over the ring $\mathbb{Z}_q[X]/(h)$ which is much smaller than the ring $\mathbb{Z}_q[X]/(f)$ of ciphertext.

We remark that the size of d_h does not increase seriously though the target computation g has more inputs and depth, and aforementioned efficiency improvement also appears in other computation. In addition, the improvement can be more drastic if the ciphertext modulus q is a power of prime bigger than 2: if $q = p^e$ for a prime p, we can take $d_h = \frac{139}{\log p}$ in our examples, resulting in additional $\times \log p$ improvement.

5 Context-Hiding VC for Nondeterministic Computations

In this section, we generalize the notion of private VC to support nondeterministic computations and public verifiability with *context-hiding*. Next, we show how to extend our construction of Section 3 to achieve these properties.

In brief, supporting nondeterministic computations means to consider functions of the form $g(x, w)$ in which the untrusted worker receives an encoding σ_x of the input x (which may hide x), holds an additional input w and can produce an encoding σ_y that, if computed honestly, is supposed to decode to $g(x, w)$. This is useful to handle more complex computations, such as ones that use random coins, non-arithmetic operations (e.g., bit-decompositions) or (secret) computation parameters. For instance, with this we can prove correct re-randomization of ciphertexts, or to evaluate polynomials with coefficients provided by the server.

For security, we require a notion similar to the soundness of proof systems, namely that a dishonest prover holding σ_x cannot produce an output σ_y which decodes to a value y for which there exists no w such that $y = g(x, w)$.

On the other hand, context-hiding—introduced in [FNP20] for deterministic computations $g(x)$ only—is a property that guarantees that the verifier does not learn any information about (x, w) beyond what can be inferred from y. Finally, public verifiability allows the computation to be verifiable by anyone (possibly a party different from the one who provided the input).

In the next section we introduce our definitions of context-hiding VC for nondeterministic computations, and then in the following section we sketch a construction of this primitive (fully detailed in the full version).

5.1 Definition of VC for Nondeterministic Computation & Context-Hiding

We extend the notion of verifiable computation from Section 2.1 to support nondeterministic computations and context-hiding, as informally explained above.

Formally, a VC scheme for non-deterministic computations is a tuple of algorithms as defined in Section 2.1 with the following differences.

$\mathsf{Compute}_{PK_g}(\sigma_x, w) \to \sigma_y$: Using the public keys, the encoded input σ_x and an additional input w, the server computes an encoded version of the function's output $y = g(x, w)$.

The notion of correctness considers the additional input w:

Correctness. For any function g and input x and w,

$$\Pr\left[\begin{array}{c} \mathsf{Verify}_{PK_g}(\tau_x, \sigma_y) = 1 \\ \wedge\ \mathsf{Decode}_{SK,SK_g}(\sigma_y) = g(x,w) \end{array} \middle| \begin{array}{r} (PK, SK) \leftarrow \mathsf{Setup}(1^\lambda) \\ (PK_g, SK_g) \leftarrow \mathsf{KeyGen}_{PK}(g) \\ (\sigma_x, \tau_x) \leftarrow \mathsf{ProbGen}_{PK}(x) \\ \sigma_y \leftarrow \mathsf{Compute}_{PK_g}(\sigma_x, w) \end{array}\right] = 1.$$

Privacy and Security. The notion of privacy is the same as in the Definition 1. For security, we instead consider the following experiment where $O_{\mathsf{KeyGen}}(g)$ is an oracle that calls $\mathsf{KeyGen}_{PK,SK}(g)$ and returns PK_g as in Section 2.1 (the difference lies in the final if condition).

Experiment $\mathbf{Exp}^{Verif}_{\mathcal{A}}[\mathcal{VC}, \lambda]$
$(PK, SK) \leftarrow \mathsf{Setup}(1^\lambda)$;
$(x, st) \leftarrow \mathcal{A}^{O_{\mathsf{KeyGen}}(\cdot)}(PK)$;
$(\sigma_x, \tau_x) \leftarrow \mathsf{ProbGen}_{PK}(x)$;
$(g, \hat{\sigma}_y) \leftarrow \mathcal{A}^{O_{\mathsf{KeyGen}}(\cdot)}(st, \sigma_x, \tau_x)$;
$acc \leftarrow \mathsf{Verify}_{PK_g}(\tau_x, \hat{\sigma}_y)$;
if $acc = 1$ and $\nexists w : \mathsf{Decode}_{SK,SK_g}(\hat{\sigma}_y) = g(x,w)$ output 1;
else output 0;

Context-Hiding. Note that, in the definition of a publicly verifiable VC scheme, anyone with PK_g and τ_x can run Verify on σ_y to verify the correctness of the computation. A party who has the secret keys SK_g, SK can additionally get the computation result y encoded in σ_y.

In some applications, however, one may want to be assured that σ_y reveals nothing beyond y. In particular, it should not leak information about the inputs (x, w). We formalize this property in the following context-hiding notion. More specifically, we are interested in modeling two cases:

(a) no information about (x, w) should be leaked to a party that has τ_x (for verification) and SK, SK_g (for decoding of σ_y) together with σ_y;
(b) no information about w should be leaked to a party that, in addition to the information above (i.e., SK, SK_g, τ_x) has σ_x.

To motivate the properties above, consider an application where Alice stores encrypted confidential data x on a server $\mathcal{P}$ and allows a user Bob to get the results of a classification algorithm (computed by $\mathcal{P}$ with its own secret parameters w) on her data. In this case, one can be interested that no information about Alice's data x and the server's parameters w are leaked to Bob from the encoding σ_y when he decodes the result, e.g., in the case of lattice-based encryption, the noise revealed during the decryption of ciphertexts exposes such information. Furthermore, there can be use cases where Alice and Bob are the same entity, in which case we want to keep w hidden even to the party that knows x and its encoding σ_x.

Definition 7 (Context Hiding). *A VC scheme is context-hiding if there exist simulator algorithms $S_\tau, S_1, S_2, S_{3,a}, S_{3,b}$ such that:*

1. *the keys* (PK, SK) *and* (PK^*, SK^*) *are statistically indistinguishable, where* $(PK, SK) \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(PK^*, SK^*, td) \leftarrow S_1(1^\lambda)$*;*
2. *for any* g *the keys* (PK_g, SK_g) *and* (PK_g^*, SK_g^*) *are statistically indistinguishable, where* $(PK_g, SK_g) \leftarrow \mathsf{KeyGen}_{PK,SK}(g)$ *and* $(PK_g^*, SK_g^*, td_g) \leftarrow S_2(td, g)$*;*
3. *for any simulated keys* $(PK^*, SK^*, td) \leftarrow S_1(1^\lambda)$, $(PK_g^*, SK_g^*, td_g) \leftarrow S_2(td, g)$, *any function* g, *any inputs* (x, w), *and any honestly generated input/output encodings* $(\sigma_x, \tau_x) \leftarrow \mathsf{ProbGen}_{PK}(x)$, $\sigma_y \leftarrow \mathsf{Compute}_{PK_g}(\sigma_x, w)$, *the following distributions are negligibly close:*
   (a) $(PK^*, SK^*, PK_g^*, SK_g^*, \tau_x, \sigma_y) \approx (PK^*, SK^*, PK_g^*, SK_g^*, \tau_x^*, \sigma_y^*)$ *where* $\tau_x^* \leftarrow S_\tau(td)$ *and* $\sigma_y^* \leftarrow S_{3,a}(td_g, \tau_x^*, g(x, w))$*;*
   (b) $(PK^*, SK^*, PK_g^*, SK_g^*, \sigma_x, \tau_x, \sigma_y) \approx (PK^*, SK^*, PK_g^*, SK_g^*, \sigma_x, \tau_x, \sigma_y^*)$ *where* $\sigma_y^* \leftarrow S_{3,b}(td_g, \tau_x, g(x, w))$.

In the following lemma we show that property (3.a) of Definition 7 can be reduced to a simpler requirement essentially saying that τ_x statistically hides x.

Lemma 5. *Let* $\mathcal{VC}$ *be a VC scheme for which there exist simulator algorithms* $S_1, S_2, S_{3,b}$ *such that properties 1,2,3.b of Definition 7 hold. Furthermore, assume there exists a simulator algorithm* S_τ *such that, for any* $(PK^*, SK^*, td) \leftarrow S_1(1^\lambda)$, *for any input* x *and* $(\sigma_x, \tau_x) \leftarrow \mathsf{ProbGen}_{PK}(x)$, *we have* $S_\tau(td) \approx \tau_x$. *Then* $\mathcal{VC}$ *satisfies Definition 7.*

Proof. We prove the lemma by constructing the simulator $S_{3,a}$ from $S_{3,b}$ and S_τ as follows. Let $S_{3,a}(td, \tau_x^*, y)$ be the algorithm that simply outputs $S_{3,b}(td, \tau_x^*, y)$. To see that property (3.a) of Definition 7 is satisfied: first, observe that property 3.*b* holds even when removing σ_x from the view; second, we can create an hybrid view in which we replace τ_x with a simulated one $\tau_x^* \leftarrow S_\tau(td)$. By the simulation property of S_τ this view is negligibly close to the previous one (i.e., that of property (3.b) without σ_x). Also, the latter view is identical to that in the right-hand side of property (3.a). □

Note that, as introduced in [FNP20], context-hiding is a meaningful property even in the case of deterministic computations, i.e., for empty w, where it assures that the values τ_x and σ_y do not reveal additional information on the input x.

5.2 Overview of Our Construction

We give an informal description of our VC construction that supports non-deterministic computations and context-hiding. For space restrictions, a detailed description of the scheme and the necessary building blocks are in the full version.

Let $g : R_t^n \times R_t^m \to R_t$ denote the nondeterministic computation to be delegated, and let $x \in R_t^n$ and $w \in R_t^m$ be the inputs of the client $\mathcal{C}$ and the prover $\mathcal{P}$ respectively. Also, assume that $\mathcal{P}$ receives an encryption $c_x = \mathsf{SHE.Enc}_{\mathsf{pk}}(x)$.

In order to compute an encryption of $g(x, w)$, $\mathcal{P}$ can first encode w in a ciphertext c_w,[16] and then perform the corresponding encrypted computation $\hat{g} : (R_q^D)^n \times (R_q^D)^m \to \mathbb{Z}_q[X]^D$ (without reduction modulo f)[17] to obtain $c_y =$

[16] This operation can be deterministic, e.g., embedding w in the ciphertext space.
[17] Recall that $R_q := \mathbb{Z}_q[X]/(f)$ and $\hat{g}(c_x, c_w) \bmod f = \mathsf{FHE.Eval}_{pk}(g, (c_x, c_w))$.

$\hat{g}(c_x, c_w)$. Note that checking the validity of such c_y means to check that

$$\exists c_w \in \Omega : c_y = \hat{g}(c_x, c_w)$$

where Ω is a (sub)set of valid ciphertexts.

Computing c_y in this way would not be enough for context-hiding, as c_y (in particular its "noise") may contain information on (x, w). To solve this issue, we exploit the noise flooding technique: $\mathcal{P}$ adds to the result $\hat{g}(c_x, c_w)$ an encryption c_0 of 0 with large noise that statistically hides the noise in $\hat{g}(c_x, c_w)$. If we let $\Omega_0 \subset \{\mathsf{SHE.Enc}_{\mathsf{pk}}(0)\}$ be a subset of encryptions of 0 with the appropriate noise level, then checking the validity of such computation means to check that

$$\exists c_w \in \Omega, c_0 \in \Omega_0 : c_y = \hat{g}(c_x, c_w) + c_0$$

For the sake of achieving context-hiding, the above statement must be verifiable without knowing c_x, as context-hiding asks for hiding x even against a party who has the decryption key SK and may thus figure out x from c_x. For this reason, we follow an approach similar to [FNP20]: the client creates a commitment com_x to c_x and gives to the verifier $\tau_x = \mathsf{com}_x$, while the prover proves that it knows (c_x, c_w, c_0) such that com_x opens to c_x, $c_w \in \Omega, c_0 \in \Omega_0$ and $c_y = \hat{g}(c_x, c_w) + c_0$.

Next, to avoid that the cost of generating the proof above depends on $O(|\hat{g}| \cdot d_f)$ we adapt the homomorphic hashing technique to this context. In our (interactive) protocol, the prover sends to the verifier the result c_y (as in Section 3) as well as commitments $(\mathsf{com}_w, \mathsf{com}_0)$ to (c_w, c_0) and proves knowledge of their opening; next the verifier picks a homomorphic hash function H; finally, the prover creates a proof that the openings (c_x, c_w, c_0) of $(\mathsf{com}_x, \mathsf{com}_w, \mathsf{com}_0)$ are such that:

$$H(c_y) = \hat{g}(H(c_x), H(c_w)) + H(c_0) \wedge c_w \in \Omega \wedge c_0 \in \Omega_0$$

Starting from this idea, we enhance it in two ways.

First, by using the commit-and-prove paradigm we split further the statement above into two statements linked by the same commitment. Namely, we let the prover commit to $(\gamma_x = H(c_x), \gamma_w = H(c_w), \gamma_0 = H(c_0))$ in $(\mathsf{com}'_x, \mathsf{com}'_w, \mathsf{com}'_0)$ and then prove the following two relations w.r.t. such commitment:

$$\mathcal{R}_H : \gamma_x = H(c_x) \wedge \gamma_w = H(c_w) \wedge \gamma_0 = H(c_0) \wedge c_w \in \Omega \wedge c_0 \in \Omega_0$$
$$\mathcal{R}_{\hat{g}} : H(c_y) = \hat{g}(\gamma_x, \gamma_w) + \gamma_0$$

With this splitting we can use two separate proof systems: one for $\mathcal{R}_{\hat{g}}$ which is the computation $\hat{g}(\cdot)$ (over the small ring $\mathcal{D}_H$), and one for $\mathcal{R}_H$ which is about correct hashing and the suitability of the committed ciphertexts c_w, c_0.

Second, by exploiting the structure of the BV HE scheme, we discuss how to encode the checks $c_w \in \Omega \wedge c_0 \in \Omega_0$ in an efficient manner. For $c_0 \in \Omega_0$, we assume that in the (trusted) key generation one generates a vector of ciphertexts $\vec{\omega}_0 = (\omega_{0,i})_{i=1}^{z}$ such that each of them is an encryption of 0. Then, c_0 can be

generated as $\langle\vec{\beta},\vec{\omega}_0\rangle$ where $\vec{\beta} \in \{0,1\}^z$ is a random binary vector. This way proving $c_0 \in \Omega_0$ boils down to proving that $\exists\vec{\beta} \in \{0,1\}^z : c_0 = \langle\vec{\beta},\vec{\omega}_0\rangle$.

For $c_w \in \Omega$, we prove the embedding of the plaintext in the ciphertext space. Namely, parsing c_w as the vector of coefficients $(c_{w,1},\ldots,c_{w,(D+1)\cdot d_f})$, we need to prove that $c_{w,i} \in (-t/2,t/2] \cap \mathbb{Z}$, for $i = 1$ to d_f, and $c_{w,i} = 0$ for all $i > d_f$.

The solution sketched above needs two commit-and-prove arguments, one for $\mathcal{R}_H$ and one for $\mathcal{R}_{\hat{g}}$. We also present a variant VC construction in which the relation $\mathcal{R}_{\hat{g}}$ can be proven using a *non-zero-knowledge* verifiable computation such as GKR. In this case, we reveal the values $(\gamma_x = H(c_x), \gamma_w = H(c_w), \gamma_0 = H(c_0))$ to the verifiers, yet we show how this can preserve context-hiding. Roughly, we prove that when c is a fresh ciphertext (and thus has enough entropy), its hash $H(c)$ does not reveal any information about it. To use this assumption we modify the VC construction so that c_w is freshly encrypted (instead of embedding a plaintext in a deterministic way), whereas for c_x we show that it can be hashed a bounded number of times without loosing information on it. This assumption can also be removed if the prover re-randomizes c_x and proves its correct re-randomization in $\mathcal{R}_H$.

Acknowledgements

Research leading to these results has been partially supported by the Spanish Government under projects SCUM (ref. RTI2018-102043-B-I00), CRYPTOEPIC (ref. EUR2019-103816), SECURITAS (ref. RED2018-102321-T) and SecuRing (ref. PID2019-110873RJ-I00), by the Madrid Regional Government under project BLOQUES (ref. S2018/TCS-4339), and by a research grant from Nomadic Labs and the Tezos Foundation. This work was also supported in part by the National Research Foundation of Korea (NRF) funded by the Korean Government (MSIT) under Grant NRF-2017R1A5A1015626.

References

AHIV17. S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. Ligero: Lightweight Sublinear Arguments Without a Trusted Setup. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, October / November 2017.

BCR+19. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent Succinct Arguments for R1CS. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019.

BCS16. E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive Oracle Proofs. In M. Hirt and A. D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, October / November 2016.

Ben81. M. Ben-Or. Probabilistic Algorithms in Finite Fields. In *22nd FOCS*, pages 394–398. IEEE Computer Society Press, October 1981.

BGV12. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In S. Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

BV11. Z. Brakerski and V. Vaikuntanathan. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, August 2011.

CCH+18. R. Canetti, Y. Chen, J. Holmgren, A. Lombardi, G. N. Rothblum, and R. D. Rothblum. Fiat-Shamir From Simpler Assumptions. Cryptology ePrint Archive, Report 2018/1004, 2018. https://eprint.iacr.org/2018/1004.

CCKP19. S. Chen, J. H. Cheon, D. Kim, and D. Park. Verifiable Computing for Approximate Computation. Cryptology ePrint Archive, Report 2019/762, 2019. https://eprint.iacr.org/2019/762.

CGGI16. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2016.

CGGI17. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 377–408. Springer, Heidelberg, December 2017.

CH18. H. Chen and K. Han. Homomorphic Lower Digits Removal and Improved FHE Bootstrapping. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 315–337. Springer, Heidelberg, April / May 2018.

CHM+20. A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.

CKKS17. J. H. Cheon, A. Kim, M. Kim, and Y. S. Song. Homomorphic Encryption for Arithmetic of Approximate Numbers. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 409–437. Springer, Heidelberg, December 2017.

CMT12. G. Cormode, M. Mitzenmacher, and J. Thaler. Practical verified computation with streaming interactive proofs. In S. Goldwasser, editor, *ITCS 2012*, pages 90–112. ACM, January 2012.

DM15. L. Ducas and D. Micciancio. FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640. Springer, Heidelberg, April 2015.

FGP14. D. Fiore, R. Gennaro, and V. Pastro. Efficiently Verifiable Computation on Encrypted Data. In G.-J. Ahn, M. Yung, and N. Li, editors, *ACM CCS 2014*, pages 844–855. ACM Press, November 2014.

FNP20. D. Fiore, A. Nitulescu, and D. Pointcheval. Boosting Verifiable Computation on Encrypted Data. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 124–154. Springer, Heidelberg, May 2020.

FV12. J. Fan and F. Vercauteren. Somewhat Practical Fully Homomorphic Encryption. Cryptology ePrint Archive, Report 2012/144, 2012. http://eprint.iacr.org/2012/144.

Gen09. C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

GGP10. 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, Heidelberg, August 2010.

GGPR13. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic Span Programs and Succinct NIZKs without PCPs. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.

GKP+13. S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. How to Run Turing Machines on Encrypted Data. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Heidelberg, August 2013.

GKR08. 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.

GSW13. C. Gentry, A. Sahai, and B. Waters. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

Kil92. J. Kilian. A Note on Efficient Zero-Knowledge Proofs and Arguments (Extended Abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

PRV12. 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, Heidelberg, March 2012.

Sho99. V. Shoup. Efficient Computation of Minimal Polynomials in Algebraic Extensions of Finite Fields. In *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation*, ISSAC '99, 1999.

SV14. N. P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *Designs, codes and cryptography*, 71(1):57–81, 2014.

Tha13. J. Thaler. Time-Optimal Interactive Proofs for Circuit Evaluation. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 71–89. Springer, Heidelberg, August 2013.

VZGG13. J. Von Zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge university press, 2013.

Wan03. Z.-X. Wan. *Lectures on finite fields and Galois rings*. World Scientific Publishing Company, 2003.

WTs+18. R. S. Wahby, I. Tzialla, a. shelat, J. Thaler, and M. Walfish. Doubly-Efficient zkSNARKs Without Trusted Setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society Press, May 2018.

XZZ+19. T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song. Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 733–764. Springer, Heidelberg, August 2019.