# Flexible and Efficient Verifiable Computation on Encrypted Data

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

[1] IMDEA Software Institute, Madrid, Spain
{ignacio.cascudo, dario.fiore}@imdea.org
[2] Department of Mathematical Sciences, Seoul National University
dwkim606@snu.ac.kr
[3] CentraleSupelec, University of Paris-Saclay, Gif-sur-Yvette, France
alexandre.bois-verdiere@student.ecp.fr

**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 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 clients 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 in encrypted form 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

⋆ Work done while at IMDEA Software Institute.

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 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 ciphertext 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:

| 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.

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 work [FGP14, FNP20], we can use the [BV11] somewhat homomorphic encryption scheme (where ciphertexts are in $\mathbb{Z}_q[X]/(f)$) *instantiated 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.

[4] Precisely, our basic scheme in Section 3 works for a $q$ that is a prime power; in appendix D we generalize it to any (possibly composite) integer $q$.

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 speedup 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.

**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 as 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.

### 2.1 Notation

Let $\lambda \in \mathbb{N}$ be the security parameter. We say that a function $F$ is *negligible* in $\lambda$ 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.

### 2.2 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* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{ProbGen}, \mathsf{Compute}, \mathsf{Verify}, \mathsf{Decode})$*:*

$\mathsf{Setup}(\lambda) \to (PK, SK)$ : *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 $\sigma_x$ that is given to the server to compute with, and a public value $\tau_x$ 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 $\sigma_x$, the server computes an encoded version $\sigma_y$ 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 $\tau_x$ for an input $x$, the verification algorithm converts the server's output $\sigma_y$ into a bit $acc$. If $acc = 1$ we say the client accepts (which means that $\sigma_y$ 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 $\sigma_y$ to some value $y$.*

*Remark 1.* In our definition we did not include $PK$ among the inputs of $\mathsf{Compute}$ and $\mathsf{Verify}$; this can be done without loss of generality as in any scheme one can include $PK$ into $PK_g$. Also, note that $\mathsf{ProbGen}$ 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 $\mathsf{KeyGen}$; 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 $x$,

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

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

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

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

### 2.3 Fully Homomorphic Encryption

We recall the definition of fully homomorphic encryption (FHE).

**Definition 2 (Fully Homomorphic Encryption).** *A* (public-key) fully homomorphic encryption scheme $\mathsf{FHE}$ *is a tuple of algorithms* $(\mathsf{FHE.ParamGen}, \mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ *defined as follows:*

$\mathsf{FHE.ParamGen}(\lambda)$ : *generates the public parameters of the scheme, such as descriptions of plaintext space $\mathcal{M}$, ciphertext space, keyspace, randomness distributions, etc. The output of* $\mathsf{ParamGen}$ *is assumed to be input to any subsequent algorithm.*

$\mathsf{FHE.KeyGen}(\lambda) \to (\mathsf{pk}, \mathsf{evk}, \mathsf{dk})$ : *outputs a public encryption key* $\mathsf{pk}$*, a public evaluation key* $\mathsf{evk}$*, and a secret decryption key* $\mathsf{dk}$*.*

$\mathsf{FHE.Enc}_{\mathsf{pk}}(m) \to c$ : *encrypts a message $m \in \mathcal{M}$ under public key* $\mathsf{pk}$*, and outputs ciphertext $c$.*

$\mathsf{FHE.Dec}_{\mathsf{dk}}(c) \to m^*$ : *decrypts a ciphertext $c$ using* $\mathsf{dk}$*, and outputs a plaintext $m^* \in \mathcal{M}$.*

$\mathsf{FHE.Eval}_{\mathsf{evk}}(g, c_1, \ldots, c_n) \to c^*$ : *Given the evaluation key* $\mathsf{evk}$*, a circuit $g : \mathcal{M}^n \to \mathcal{M}$, and a set of $n$ ciphertexts $c_1, \ldots, c_t$, it computes an output ciphertext $c^*$.*

A FHE scheme satisfies *correctness* and *security* as follows:

**Correctness**. For all $(m_i)_{i=1}^n \in \mathcal{M}^n$ and any (admissible) circuit $g : \mathcal{M}^n \to \mathcal{M}$, we have:

$$\Pr\left[\begin{array}{c}(\mathsf{pk}, \mathsf{evk}, \mathsf{dk}) \xleftarrow{\$} \mathsf{FHE.KeyGen}(\lambda) \\ c_i \xleftarrow{\$} \mathsf{FHE.Enc}_{\mathsf{pk}}(m_i)\ (i \in \{1, 2, \ldots, n\})\end{array} : \begin{array}{r}\mathsf{FHE.Dec}_{\mathsf{dk}}(\mathsf{FHE.Eval}_{\mathsf{evk}}(g, c_1, \ldots, c_n)) \\ = g(m_1, m_2, \ldots, m_n)\end{array}\right] = 1.^{5}$$

**Semantic Security**. For any PPT adversary $\mathcal{A}$ and $m_0 \neq m_1 \in \mathcal{M}$, it holds that:

$$\Pr\left[\begin{array}{c}(\mathsf{pk}, \mathsf{evk}, \mathsf{dk}) \xleftarrow{\$} \mathsf{FHE.KeyGen}(\lambda) \\ b \xleftarrow{\$} \{0,1\} \\ c_b \xleftarrow{\$} \mathsf{FHE.Enc}_{\mathsf{pk}}(m_b)\end{array} : \mathcal{A}(\mathsf{pk}, \mathsf{evk}, c_b) = b\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

**Compactness**. 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. Formally, there exists some polynomial $p$ such that, for any $(\mathsf{pk}, \mathsf{evk}, \mathsf{dk}) \xleftarrow{\$} \mathsf{FHE.KeyGen}(\lambda)$, the output size of $\mathsf{FHE.Enc}_{\mathsf{pk}}$ and of $\mathsf{Eval}_{\mathsf{evk}}$ is bounded by $p$, for any choice of their inputs.

In our work, we mainly consider a restricted notion of $\mathsf{SHE}$ scheme (a.k.a. somewhat homomorphic) that guarantees correctness only if the circuit $g$ is of a bounded degree which is fixed a-priori in $\mathsf{SHE.ParamGen}$. The notion of compactness is also relaxed so that the size of ciphertexts depends on the degree bound.

### 2.4 Succinct Argument Systems

We recall the definition of (succinct) argument system.

**Definition 3 (Argument System for $\mathcal{R}$).** *Let $\mathcal{R}$ be an NP relation. An argument system $\Pi$ for $\mathcal{R}$ comprises three algorithms $(\Pi.\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ as follows:*

[5] Most FHE schemes have $1 - \mathsf{negl}(\lambda)$ probability, but we assume the probability 1 for ease of exposition.

$\Pi.\mathsf{Setup}(\lambda) \to \mathsf{crs}$ : *on input a security parameter* $\lambda$*, outputs a common reference string* crs*.*

$\mathcal{P}(crs, x, w)$ : *an algorithm called prover takes as input* crs*, a statement* $x$*, and a witness* $w$*, then interacts with a verifier (in the following).*

$\mathcal{V}(crs, x)$ : *an algorithm called verifier takes* crs *and the statement* $x$*, then outputs 0 (reject) or 1 (accept) after interacting with the prover.*

An execution between the prover $\mathcal{P}$ and verifier $\mathcal{V}$ is denoted by $\langle \mathcal{P}(crs, x, w), \mathcal{V}(crs, x)\rangle_{\Pi} = b$, where $b \in \{0, 1\}$ is the output of the verifier after the interaction. If $\mathcal{V}$ uses public randomness only, we say that the protocol is *public-coin*. An argument system is *interactive* with $\lceil \frac{k}{2} \rceil$ rounds if $\mathcal{P}$ and $\mathcal{V}$ exchange $k \geq 2$ messages before $\mathcal{V}$ accepts or rejects (otherwise, it is called non-interactive).

An argument system $\Pi$ satisfies *completeness*, *soundness*, and *succinctness* as defined below.

**Completeness**. For all $(x, w)$ such that $\mathcal{R}(x, w) = 1$, we have

$$\Pr\left[\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(\lambda) : \langle \mathcal{P}(crs, x, w), \mathcal{V}(crs, x)\rangle_{\Pi} = 1\right] = 1.$$

**Soundness**. For all PPT prover $\mathcal{P}^* = (\mathcal{P}_0^*, \mathcal{P}_1^*)$ we have

$$\Pr\left[\begin{array}{c} \langle \mathcal{P}_1^*(\mathsf{crs}, x, st), \mathcal{V}(\mathsf{crs}, x)\rangle_{\Pi} = 1 \\ \wedge\, \nexists w \text{ s.t. } \mathcal{R}(x, w) = 1 \end{array} \middle| \begin{array}{l} \mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(\lambda) \\ (x, st) \leftarrow \mathcal{P}_0^*(\mathsf{crs}) \end{array}\right] \leq \delta.$$

Note that the above soundness condition corresponds to the *adaptive soundness* of [FGP14] in which the adversary is allowed to choose the statement $x$ *after* seeing the reference string crs.

**Succinctness**. The communication cost and the running time of a verifier $\mathcal{V}$ in the protocol execution are asymptotically dominated by the size of $w$ and the time required to compute $\mathcal{R}(x, w)$, respectively (considering the security parameter $\lambda$ as a constant).

Note, in above definitions, that the algorithm $\Pi.\mathsf{Setup}$ must be performed by a trusted party (or a verifier) and is necessary in non-interactive case.

**Preprocessing Arguments with Universal CRS**. In our VC construction, we also consider a variant that uses a preprocessing succinct argument with a universal CRS (we refer to [CHM+20] for formal definitions). In brief, these are argument systems in which the algorithm $\Pi.\mathsf{Setup}$ generates a universal reference string CRS that works for *a family of NP relations*, as opposed to a single one. In there, a deterministic algorithm Preprocess, on input the universal CRS and a relation $\mathcal{R}$, outputs a $\mathsf{crs}_{\mathcal{R}}$ which substitutes the crs in the above definition. Notably, the $\mathsf{crs}_{\mathcal{R}}$ comprises a succinct encoding of the relation (e.g., a circuit representation) and is useful to boost efficiency for the verifier.

## 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 $\mathsf{SHE}$ 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[6] 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 $\varepsilon$-universal whose definition follows.

**Definition 4 ($\varepsilon$-Universal Hash Functions).** *A family $\mathcal{H}$ of hash functions is $\varepsilon$-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 $\Pi$ (Definition 3) 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 2.2, except for $\mathsf{Compute}$ and $\mathsf{Verify}$ 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.

In our VC scheme, a verifier $\mathcal{V}$ encrypts the input $x = (x_1, x_2, \ldots, x_n)$ (with $\mathsf{SHE}$) 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

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

$f$, 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.

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

$\mathsf{KeyGen}_{PK}(g) \to (PK_g, SK_g)$:
- Run $\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(\lambda)$ to generate the common reference string of $\Pi$ 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 $c_y = \hat{g}(c_x)$ (without reduction modulo $f$)and sends it to $\mathsf{Verify}$.
- $\mathsf{Verify}$ samples and sends a homomorphic hash function $H \xleftarrow{\$} \mathcal{H}$.
- $\mathsf{Compute}$ and $\mathsf{Verify}$ both compute $\gamma_1 = H(c_1), \ldots, \gamma_n = H(c_n), \gamma_y = H(c_y)$.
- $\mathsf{Compute}$ and $\mathsf{Verify}$ 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 $\sigma_y$ include $c_y$ and the transcript of the interactive argument.
- Let $b$ be the bit returned by $\mathcal{V}$. $\mathsf{Verify}$ 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 $\Pi$ 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 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 $\Pi$. Then, $\mathsf{Compute}$ can be described as a non-interactive algorithm that on input $\sigma_x$ outputs $\sigma_y = (c_y, \pi)$ where $\pi$ 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., $\mathcal{V}(PK_g, H(c_1), \ldots, H(c_n), H(c_y))$.

*Remark 3 (On a variant using universal arguments).* Note that, if $\Pi$ is a preprocessing argument system with a *universal* CRS [CHM+20], then one can modify our VC scheme as follows: $\Pi.\mathsf{Setup}$ can be executed in $\mathsf{Setup}$ (of VC scheme) and the universal CRS is included in $PK$, while $\mathsf{KeyGen}$ would only run the deterministic preprocessing $\mathsf{crs}_g \leftarrow \mathsf{Proprocess}(\mathrm{CRS}, g)$. The main benefit of this variant is that only the $\mathsf{Setup}$ 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 $\lambda$, if we exploit a correct, compact, and secure* SHE *scheme, an $\varepsilon$-universal family of hash functions with $\varepsilon = \mathsf{negl}(\lambda)$, and a complete succinct argument system $\Pi$ 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 2.2. 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 $\Pi$.

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 $\Pi$ 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 2.2, adapted to handle the case of protocols in which Compute and Verify interact. Namely, instead of an adversary that directly provides the result $\sigma_y$, 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 $\varepsilon$-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 $\Pi$ with the false result $H(\hat{c}_y)$. Let us note $B$ the event $\{\Pi(\mathcal{V}, \mathcal{A}, H(c), H(\hat{c}_y), r) = 1\}$. If $\Pi$ has soundness $\delta$ 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 $\varepsilon$ and $\delta$ are $\mathsf{negl}(\lambda)$. □

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

**Lemma 1.** *Let $T_{\mathcal{P}}^{\Pi}, T_{\mathcal{V}}^{\Pi}$, and $C^{\Pi}$ respectively denote the time complexity of $\mathcal{P}$, that of $\mathcal{V}$, and the communication cost in the argument system $\Pi$, which will be signified as, e.g., $T_{\mathcal{P}}^{\Pi}(g; R)$ when denoting the complexity of $\mathcal{P}$ (in $\Pi$) 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 time 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).

*Remark 4.* The space complexity of $\mathsf{Verify}_{PK_g}(\tau_x)$ is also improved in our VC, since it is the same as the space complexity of $\mathcal{V}$ in $\Pi$ for a circuit $\hat{g}$ over $\mathcal{D}_H$, given that the evaluation of homomorphic hash function is space-efficient (as will be shown in Section 4.4). Here, we only mention that $\mathcal{V}$ does not need to store the result $\hat{g}(c)$ and only needs to store the output of hash evaluation which is much smaller.

**Applications to VC for SIMD Computations**. Besides the application to HE computations, the use of $\varepsilon$-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 $|\hat{g}| \approx |g| \cdot d_f$.[7] 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 detailed efficiency analysis with example parameters.

[7] 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.

### 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.[8] 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:

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

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 $R_q[Y]$:

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

The *correctness* ($\mathsf{Dec}(c_{\text{add}}) = m + m'$, $\mathsf{Dec}(c_{\text{mult}}) = m \cdot m'$) is guaranteed only if the degree (in $Y$) of result ciphertext ($c_{\text{add}}$ or $c_{\text{mult}}$) 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 Appendix A for the description of other algorithms ($\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$) of BV scheme and the concrete conditions (Lemma 7) 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[9] 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].

[8] 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.

[9] 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.

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., $\mathsf{KeyGen}_{PK}(g)$ founds such description for $g$ and put it into $PK_g$). We remark that, in our instantiation, the target circuit already has low depth to be supported by $\mathsf{BV}$ scheme.

In this section, we recall the definition of Galois rings, the Schwartz-Zippel lemma for rings, and give a summary on the GKR protocol over rings. See Appendix B for 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].

**Definition 5 (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 6 (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 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-coin) 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, the complexity of our instantiation is also $(T_{\mathcal{P}}^{\Pi}, T_{\mathcal{V}}^{\Pi}, C^{\Pi})$[10] $= (O(S),\ O(n + d\log S)$[11]$,\ O(d\log S))$. 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 $\varepsilon$-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 work [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 7 (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 $\epsilon$-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 $\varepsilon$-universal for $\varepsilon = \frac{2N}{p^{d_{\mathcal{H}}}}$.*

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

[11] 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.

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

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

*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, $\Delta$ is a non zero polynomial of degree at most $N$, and $H_h(z) = H_h(z')$ implies that $\Delta$ has $h \in \mathbb{Z}_q[X]$ as a factor, which is equivalent to that $\Delta$ has $h$ as a factor in $\mathbb{Z}_p[X]$ (after modulo reduction by $p$).[12] 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 $\mu$ be the Möbius function,[13] $p$ 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 $\mu$, 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 5 (Setting $\varepsilon$).* We can set $\varepsilon$ 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, the degree $N$ of polynomials in $\mathcal{D}$ is bounded by $(d_f - 1)(D - 1)$, and $\varepsilon$ can be bounded by $\frac{2(d_f-1)(D-1)}{p^{d_{\mathcal{H}}}}$.

*Remark 6 (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.

[12] More precisely, the argument follows if $\Delta$ is not zero when reduced modulo $p$. Otherwise, $\Delta = p^k \delta$ for some $k < e$ and a polynomial $\delta$ which is not zero when reduced modulo $p$, and $\delta$ 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 $\delta$ in $\mathbb{Z}_p[X]$.

[13] 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, $\mu(n) = 0$.

### 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 8 (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 7) 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 $\epsilon$-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 6) *of $\mathbb{Z}_q[X]/(h)\}$ is homomorphic and $\varepsilon$-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 7) 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, $\Delta$ 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 6) 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 $\varepsilon$ 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 7 (Comparison to Single Hash).* Utilizing double hashes instead of single hashes give 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 8 (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 6) 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 $\leq 2^{-\lambda}$.*

We refer to Appendix E.1 for the 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 $\alpha$ of the multiplicative group $\mathbb{F}^{\times}_{p^{d_h}}$, these sampleable elements are exactly $\alpha^j$ 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[14]

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

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

**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 $\alpha$ 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 $\beta$, using the algorithm in [Sho99].
  5. Return $h$

*Proof.* See Appendix E.2 for the proof. □

*Remark 9.* 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 10.* 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 Instantiation

The aggregation of building blocks can be summarized as follows:

**SHE - BV scheme:** In setup, take a polynomial $f$ of (large enough) degree $d_f$ and a ciphertext modulus $q$ satisfying $2^\lambda$ security and the correctness condition (Appendix A, Lemma 7) with given degree bound $D$ and a plaintext modulus $t$, which sets the ciphertext space $R_q^D = (\mathbb{Z}_q[X]/(f))^D$.

**Homomorphic Hash:** Note that the ciphertext space $R_q^D$ can be identified as (a subset of) $\mathbb{Z}_q[X]^D$ or $\mathbb{Z}_q[X][Y]$.[15] 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).

### 4.4.2 Complexity of 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 $(T_{\mathcal{P}}, T_{\mathcal{V}}, C_{\mathcal{C}})$[16] measured by the number of operations (or elements) of $\mathbb{Z}_q$ is as follows.*

*(i) with Single hash:*

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

*(ii) with Double hash:*

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

*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 $O(D^2 S)$ 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 $O(D^2 d_f)$ 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 Appendix E.3 for the full proof. □

[15] For this, we skip the modulo reduction by $f$ at the (delegated) computation.
[16] Each signifies the time complexity of $\mathcal{P}$, that of $\mathcal{V}$, and the communication cost.

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 11 (Ciphertext Computation vs Proof Generation).* Since $d_f = \Omega(\sqrt{\lambda} \log q)$ for the security of BV scheme (from the hardness of LWE, Appendix C), 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.

*Remark 12 (Space Complexity of $\mathcal{V}$).* Our instantiation of VC supports space-efficient verification as does the GKR protocol: with single or double hash, the space complexity of verifier is $O(\lambda \log (D^2 S))$ or $O(\lambda \log S)$, respectively. Note that computing the remainder of polynomial division (required in evaluation of hash) can be done using the space proportional to the degree of divisor polynomial only, if the dividend polynomial is given with appropriate streaming.

### 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 assume that the ciphertext modulus $q$ and the degree $d_f$ of $f$ satisfies the following equation for the correctness (Lemma 7, Section 4) of BV scheme.

$$||g||_\infty (t \cdot \sigma \cdot d_f^{1.5})^{(D-1)} \leq \frac{q}{2} \tag{2}$$

On the other hand, for the security of BV scheme, underlying $\mathsf{PLWE}_{f,q,\chi}$ problem or corresponding $\mathsf{LWE}_{d_f,q,\alpha(=\frac{8}{q})}$ problem[17] should provide $2^\lambda$ security, which can be simplified as the following condition when $\lambda = 128$ (see Appendix C).

$$\log q \leq \frac{d_f}{30} + 1, \tag{3}$$

which, combined with the equation (2) and $||g||_\infty \leq t$ and $\sigma \approx 3.2 \leq 4$, gives the following

$$D \log t + (D-1)(2 + \frac{3}{2} \log d_f) + 1 \leq \log q \leq \frac{d_f}{30} + 1 \tag{4}$$

On the other hand, for the $2^{-\lambda}$ soundness of our scheme (from the $2^{-(\lambda+1)}$ universality of homomorphic hash functions and the $2^{-(\lambda+1)}$ soundness of GKR protocol), the degree $d_h$ of $h$ should satisfy that (see Theorem 2, Theorem 3,4, and Theorem 7),

(i) when using single hash,

$$\begin{aligned} 2^{\lambda+1} \cdot 7d \log(D^2 S) &\leq p^{d_h} \\ 2^{\lambda+2} \cdot (d_f - 1)(D-1) &\leq p^{d_h} \end{aligned} \tag{5}$$

[17] Here, we regard given PLWE problem as a corresponding LWE problem as usual.

(ii) when using double hash,

$$\begin{aligned} 2^{\lambda+1} \cdot 7d \log S &\leq p^{d_h} \\ 2^{\lambda+1} \cdot (2(d_f - 1)(D-1) + D - 1) &\leq p^{d_h} \end{aligned} \tag{6}$$

where we assume that $q = p^e$ for a prime $p$. From these equations, we can set the parameters of our scheme in the following examples. On the security of BV scheme, we used the estimator [APS15] (`https://bitbucket.org/malb/lwe-estimator`) rather than equation (8) for more accurate analysis.

**Example I. Inner Product**.
Assume a verifier delegates the inner product of two $n$-dimensional vectors over $R_t = \mathbb{Z}_t[X]/(f)$, which can be represented by the computation of following function:

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

Since the function $g$ is of multiplicative depth 1, it suffices to take $D = 3$ in the parameters of BV scheme. The circuit description of $g$ is simple: it takes $2n$ inputs, has $n$ multiplication gates (each computing $a_i \times b_i$), and a tree of addition gates summing $n$ terms of $a_i b_i$'s. Since the tree of addition gates can be efficiently handled with GKR protocol, we regard them as one layer consisting of $n$ addition gates when setting parameters for the protocol. Taking these into account, the parameters for our scheme with double hash satisfying equation (4) and (6) can be found in Table 2. We assume that the plaintext modulus $t$ is a prime of 8-bit or 16-bit, and the ciphertext modulus $q$ is a power of two. We also considered a more tight bound than equation (4) since $||g||_\infty = 1$.

**Example II. Polynomial Evaluation**.
As the second example, assume a verifier delegates a parallel evaluation of a polynomial $F \in R_t[Y]$ of degree $d_F$ on $n$ elements over $R_t = \mathbb{Z}_t[X]/(f)$, which can be represented by the computation of following function (we denote the coefficients of $F$ as $F_i$'s).

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

The function $g$ is of degree $d_F + 1$, and it suffices to take $D = d_f + 2$ in the parameters of BV scheme. For the circuit description of $g$, we assume that the evaluation of polynomial $F$ is done by Horner's method[18]. The method requires $d_F$ additions and $d_F$ multiplications to evaluate $F$ on one input, resulting in a total of $nd_F$ additions and $nd_F$ multiplications in the computation of $g$, and the depth of the circuit is $d_F$. Taking these into account, the parameters for our scheme with double hash satisfying equation (4) and (6) can be found in Table 2. We assume that the plaintext modulus $t$ is a prime of 2-bit or 16-bit, and the ciphertext modulus $q$ is a power of two.

**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

[18] i.e., $F(x)$ is evaluated by $F_0 + x\Big(F_1 + x\big(F_2 + \cdots + x(F_{n-1} + x\, F_n)\cdots\big)\Big)$.

| Circuit | | | BV scheme | | | | GKR & Hash | Security | |
|---|---|---|---|---|---|---|---|---|---|
| $g$ | $n$ | $d$ | $D$ | $\log t$ | $\log q$ | $\log d_f$ | $d_h$ | $\lambda_{\mathsf{BV}}$ | $\lambda_s$ |
| Inn Prod. | $2^8$ | 2 | 3 | 8 | 54 | 11 | 136 | | 128.4 |
| Inn Prod. | $2^8$ | 2 | 3 | 16 | 73 | 12 | 136 | $\geq 128$ | 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 | $\geq 128$ | 128.7 |
| Poly Eval. | $2^{12}$ | 16 | 18 | 16 | 685 | 15 | 138 | | 128.6 |

**Table 2.** Example parameters for our VC scheme: $\lambda_{\mathsf{BV}}$ and $\lambda_s$ respectively denote the bit security of BV scheme and our VC scheme.

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 $\sigma_x$ of the input $x$ (which may hide $x$), holds an additional input $w$ and can produce an encoding $\sigma_y$ 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 $\sigma_x$ cannot produce an output $\sigma_y$ 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 sections we present two constructions of this primitive.

### 5.1 Definition of VC for Nondeterministic Computation & Context-Hiding

We extend the notion of verifiable computation from Section 2.2 to support non-deterministic 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 2.2 with the following differences.

$\mathsf{Compute}_{PK_g}(\sigma_x, w) \to \sigma_y$: Using the public keys, the encoded input $\sigma_x$ 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 $w$,

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

**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 2.2 (the difference lies in the final if condition).

Experiment $\mathbf{Exp}^{Verif}_{\mathcal{A}}[\mathcal{VC}, \lambda]$
$(PK, SK) \leftarrow \mathsf{Setup}(\lambda)$;
$(x, st) \leftarrow \mathcal{A}^{O_{\mathsf{KeyGen}}(\cdot)}(PK)$;
$(\sigma_x, \tau_x) \leftarrow \mathsf{ProbGen}_{PK}(x)$;
$(g, \hat{\sigma}_y) \leftarrow \mathcal{A}^{O_{\mathsf{KeyGen}}(\cdot)}(st, \sigma_x, \tau_x)$;
$acc \leftarrow \mathsf{Verify}_{PK_g}(\tau_x, \hat{\sigma}_y)$;
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 $\tau_x$ can run $\mathsf{Verify}$ on $\sigma_y$ 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 $\sigma_y$.

In some applications, however, one may want to be assured that $\sigma_y$ 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 $\tau_x$ (for verification) and $SK, SK_g$ (for decoding of $\sigma_y$) together with $\sigma_y$;
(b) no information about $w$ should be leaked to a party that, in addition to the information above (i.e., $SK, SK_g, \tau_x$) has $\sigma_x$.

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 $\sigma_y$ 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 $\sigma_x$.

**Definition 9 (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 9 can be reduced to a simpler requirement which essentially says that $\tau_x$ 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 9 hold. Furthermore, assume there exists a simulator algorithm $S_\tau$ 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 9.*

*Proof.* We prove the lemma by constructing the simulator $S_{3,a}$ from $S_{3,b}$ and $S_\tau$ 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 9 is satisfied: first, observe that property 3.*b* holds even when removing $\sigma_x$ from the view; second, we can create an hybrid view in which we replace $\tau_x$ with a simulated one $\tau_x^* \leftarrow S_\tau(td)$. By the simulation property of $S_\tau$ this view is negligibly close to the previous one (i.e., that of property (3.b) without $\sigma_x$). 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 $\tau_x$ and $\sigma_y$ do not reveal additional information on the input $x$.

### 5.2 Overview of Our Construction

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 $c_w$,[19] 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 $f$)[20] to obtain $c_y = \hat{g}(c_x, c_w)$. 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 $\Omega$ 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 $\mathsf{com}_x$ 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 $\mathsf{com}_x$ 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 $c_w, c_0$.

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

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 be also removed if the prover re-randomizes $c_x$ and proves its correct re-randomization in $\mathcal{R}_H$.

### 5.3 Building Blocks

To sum up, our VC scheme for nondeterministic computation and context-hiding requires as additional building blocks: a somewhat homomorphic encryption that, in addition to the properties defined in Section 3, supports circuit privacy via noise flooding; a commitment scheme, and succinct zero-knowledge arguments that are commit-and-prove ($\mathsf{CaP}$). We give below the necessary definitions of these building blocks.

**Noise-flooding-based circuit-private** $\mathsf{SHE}$. Let $\mathsf{SHE}$ be a somewhat homomorphic encryption scheme with the same structure as defined in Section 3. In addition we assume that $\mathsf{SHE}$ achieves circuit privacy by re-randomizing any ciphertext $c$ output of $\mathsf{SHE.Eval}$ by adding a random linear combination of $z$ publicly available and properly generated encryptions of 0. A bit more in detail, we assume there is a sufficiently large $z = \mathsf{poly}(\lambda)$, and two simulator algorithms $\mathsf{SHE}.S_0, \mathsf{SHE}.S_1$ such that: $\mathsf{SHE}.S_0(\mathsf{pk})$ generates a vector $\vec{\omega}_0$ such that for all $i \in \{1, \ldots, z\}$, $\omega_{0,i} \in \mathsf{SHE.Enc}(0)$, and for any input $x$, any admissible function $g$, and $c \leftarrow \mathsf{Enc}(x)$, it holds $\{\hat{g}(c) + \langle \vec{\beta}, \vec{\omega}_0 \rangle : \vec{\beta} \xleftarrow{\$} \{0,1\}^z\} \approx \mathsf{SHE}.S_1(\mathsf{pk}, g(x))$.

**Commitments**. We recall the definition of (extractable) non-interactive commitment schemes.

**Definition 10 (Commitment).** *A commitment scheme is a tuple of algorithms* $Com = (\mathsf{ComGen}, \mathsf{Commit}, \mathsf{ComVer}, \mathsf{OpenVer})$*:*

$\mathsf{ComGen}(1^\lambda) \to \mathsf{ck}$**:** *generates a commitment public key, a message space* $M_{\mathsf{ck}}$*, a randomness (opening) space* $R_{\mathsf{ck}}$ *and a commitment space* $C_{\mathsf{ck}}$*. It should be run by a trusted party.*

$\mathsf{Commit}(\mathsf{ck}, m) \to (\mathsf{com}, o)$**:** $\mathcal{V}$ *given a message* $m \in M_{\mathsf{ck}}$*, outputs a commitment* $\mathsf{com} \in C_{\mathsf{ck}}$ *and an opening information* $o \in R_{\mathsf{ck}}$*.*

$\mathsf{ComVer}(\mathsf{ck}, \mathsf{com}) \to 0/1$**:** *outputs* 1 *if* $c$ *is well-formed and* 0 *otherwise.*

$\mathsf{OpenVer}(\mathsf{ck}, \mathsf{com}, m, o) \to 0/1$**:** *outputs* 1 *if* $m \in M_{\mathsf{ck}}$ *is the message in the commitment* com *and* 0 *if* $(m, o, \mathsf{com})$ *does not correspond to a valid pair opening-commitment.*

We say that a commitment $Com$ is secure if it satisfies the following properties:

**Correctness**. If $m \in M_{\mathsf{ck}}$ is honestly generated and $(\mathsf{com}, o) = \mathsf{Commit}(\mathsf{ck}, m)$ then $\mathsf{ComVer}(\mathsf{ck}, \mathsf{com}) = 1$ and $\mathsf{OpenVer}(\mathsf{ck}, \mathsf{com}, m, o) = 1$.

**Hiding**. For any two messages $m_0, m_1 \in M_{\mathsf{ck}}$, it is hard for an adversary $\mathcal{A}$ to distinguish between their corresponding commitments $\mathsf{com}_0, \mathsf{com}_1$, where $(\mathsf{com}_0, o_0) \leftarrow \mathsf{Commit}(\mathsf{ck}, m_0)$ and $(\mathsf{com}_1, o_1) \leftarrow \mathsf{Commit}(\mathsf{ck}, m_1)$.

**Binding**. It is computationally hard for any adversary $\mathcal{A}$ to find $(\mathsf{com}, m_0, o_0, m_1, o_1)$ such that $o_0$ and $o_1$ are valid opening values for two distinct messages $m_0 \neq m_1$ for $\mathsf{com}$. Namely, for any PPT $\mathcal{A}$, the following probability is negligible:

$$\Pr\left[\begin{array}{c} \mathsf{OpenVer}(\mathsf{ck}, \mathsf{com}, m_0, o_0) = 1 \\ \wedge\ \mathsf{OpenVer}(\mathsf{ck}, \mathsf{com}, m_1, o_1) = 1 \\ \wedge\ m_0 \neq m_1 \end{array} \middle| \begin{array}{c} \mathsf{ck} \leftarrow \mathsf{ComGen}(1^\lambda) \\ (\mathsf{com}, m_0, o_0, m_1, o_1) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right]$$

**Knowledge Binding**. For any adversary $\mathcal{A}$ that produces a valid commitment $c$ associated to a message, i.e. such that $\mathsf{ComVer}(\mathsf{ck}, \mathsf{com}) = 1$, there is an extractor $\mathsf{Ext}_{\mathcal{A}}$ that returns $m$ and a valid opening $o$ of $\mathsf{com}$ that is valid with overwhelming probability, i.e.,:

$$\Pr\left[\begin{array}{c} \mathsf{OpenVer}(\mathsf{ck}, \mathsf{com}, m, o) = 1 \\ \wedge \mathsf{ComVer}(\mathsf{ck}, \mathsf{com}) = 1 \end{array} \middle| \begin{array}{c} \mathsf{ck} \leftarrow \mathsf{ComGen}(1^\lambda) \\ (\mathsf{com}, (m, o)) \leftarrow (\mathcal{A} || \mathsf{Ext}_{\mathcal{A}})(\mathsf{ck}) \end{array}\right] = 1 - \mathsf{negl}(\lambda)$$

For simplicity we often omit the parameter $\mathsf{ck}$ and use $\mathsf{Commit}(m) \rightarrow (\mathsf{com}, o)$. Also, we observe that an extractable commitment can be realized by augmenting a regular commitment scheme with a proof of knowledge, i.e., the commitment includes both the commitment and the proof of knowledge, and $\mathsf{ComVer}$ consists in verifying this proof.

We recall the definition of arguments of knowledge, which is like the one in Definition 3 (in Section 2.4) except that it is required to satisfy the knowledge-soundness and the zero-knowledge property defined below.

**Knowledge-soundness**. An argument system $\Pi$ for an NP relation $\mathcal{R}(u, w)$ is *knowledge-sound* if for every $\mathsf{PPT}$ adversary $\mathcal{A}$ there exists a PPT extractor $\mathsf{Ext}_{\mathcal{A}}$ such that, for all adversaries $\mathcal{A}_0$ it holds

$$\Pr\left[\begin{array}{c} \langle \mathcal{A}_1(\mathsf{crs}, u, \mathsf{st}), \mathcal{V}(\mathsf{crs}, u)\rangle_\Pi = 1 \\ \wedge\ \mathcal{R}(u, w) = 0 \end{array} \middle| \begin{array}{c} \mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\lambda) \\ (u, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{crs}) \\ w \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathsf{crs}, u, \mathsf{st}) \end{array}\right] = \mathsf{negl}(\lambda).$$

**Honest-Verifier Zero-knowledge**. $\Pi$ is a (statistical) honest-verifier zero-knowledge (HVZK) argument for a relation $\mathcal{R}$ if there exists a stateful PPT simulator $\mathsf{Sim}$ such that for all interactive PPT distinguishers $(\mathcal{D}_0, \mathcal{D}_1)$, the following two probabilities are negligibly close:

$$\Pr\left[\begin{array}{c} \mathcal{R}(u, w) = 1 \\ \wedge\ \mathcal{D}_1(\mathsf{tr}) = 1 \end{array} \middle| \begin{array}{c} \mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\lambda) \\ (u, w, \mathsf{st}) \leftarrow \mathcal{D}_0(\mathsf{crs}) \\ \mathsf{tr}\langle \mathcal{P}(\mathsf{crs}, u, w), \mathcal{V}(\mathsf{crs}, u, \mathsf{st})\rangle_\Pi \end{array}\right] \approx \Pr\left[\begin{array}{c} \mathcal{R}(u, w) = 1 \\ \wedge\ \mathcal{D}_1(\mathsf{tr}) = 1 \end{array} \middle| \begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}(1^\lambda) \\ (u, w, \mathsf{st}) \leftarrow \mathcal{D}_0(\mathsf{crs}) \\ \mathsf{tr} \leftarrow \mathsf{Sim}(\mathsf{crs}, \mathsf{td}, u) \end{array}\right]$$

Finally, the definition of Commit-and-Prove argument ($\mathsf{CaP}$-$\Pi$) follows. It is an argument system regarding the relations on the committed messages.

**Definition 11 (Commit-and-Prove Argument).** *Let $\mathcal{R}(u,w)$ be a* NP *relation where $u = (m_i)_i$. A Commit-and-Prove argument (*CaP*) for a commitment scheme Com and relation $\mathcal{R}$ is an argument system for the "commit-and-prove" relation $\mathcal{R}^{\mathsf{ck}}(u^*,w^*)$ where $u^* = (u,(\mathsf{com}_i)_i), w^* = (\{m_i\}_i,\{o_i\}_i,w)$ and that holds if and only if $\mathcal{R}(u,w)$ holds and $\mathsf{OpenVer}(\mathsf{com}_i,m_i,o_i)=1$ for all $i$.*

### 5.4 The Generic Scheme

Before introducing our generic VC scheme for nondeterministic computation with context hiding, we present the NP relations that need to be proven in our construction.

Let $g : R_t^n \times R_t^m \to R_t$ be the computation to be delegated, and let $\hat{g} : (R_q^D)^n \times (R_q^D)^m \to \mathbb{Z}_q[X]^D$ be the corresponding computation over ciphertexts such that:

$$\hat{g}(c_{x,1},\ldots,c_{x,n},c_{w,1},\ldots,c_{w,m}) \mod f = \mathsf{FHE.Eval}_{\mathsf{evk}}(g,c_{x,1},\ldots,c_{x,n},c_{w,1},\ldots,c_{w,m}) \in R_q^D.$$

Let *Com* be a commitment scheme with key $\mathsf{ck}$. Then we define the following two relations $\mathcal{R}_H$ and $\mathcal{R}_{\hat{g}}$.

$$\begin{aligned}\mathcal{R}_H(u_H,v_H)=1 \iff & \vec{\gamma}_x = H(c_x) \wedge \vec{\gamma}_w = H(c_w) \wedge \gamma_0 = H(c_0)\\ & \wedge c_0 = \langle \vec{\beta},\vec{\omega}_0\rangle \wedge \vec{\beta} \in \{0,1\}^z\\ & \wedge c_w \in \Omega^m\\ & \wedge \mathsf{OpenVer}(\mathsf{ck},\mathsf{com}_x,(c_{x,1},\ldots,c_{x,n}),o_x) \wedge \mathsf{OpenVer}(\mathsf{ck},\mathsf{com}_w,(c_{w,1},\ldots,c_{w,m}),o_w)\\ & \wedge \mathsf{OpenVer}(\mathsf{ck},\mathsf{com}_0,c_0,o_0) \wedge \mathsf{OpenVer}(\mathsf{ck},\mathsf{com}_\gamma,(\vec{\gamma}_x,\vec{\gamma}_w,\gamma_0),o_\gamma)\end{aligned}$$

where $(u_H,v_H) = ((\vec{\omega}_0,H,\mathsf{com}_x,\mathsf{com}_w,\mathsf{com}_0,\mathsf{com}_\gamma),(c_x,c_w,c_0,(\vec{\gamma}_x,\vec{\gamma}_w,\gamma_0),o_x,o_w,o_0,o_\gamma,\vec{\beta}))$, $\vec{\omega}_0$ is a vector of encryptions of 0, and $c_w \in \Omega^m$ is a shorthand for checking that each $c_{w,i} \in R_t \times \{0\}^{D-1} \subset R_t^D$.

$\mathcal{R}_{\hat{g}}$ is over pairs $(u_{\hat{g}},v_{\hat{g}}) = ((\gamma_y,\mathsf{com}_\gamma),(\vec{\gamma}_x,\vec{\gamma}_w,\gamma_0,o_\gamma))$ and is such that:

$$\begin{aligned}\mathcal{R}_{\hat{g}}(u_{\hat{g}},v_{\hat{g}})=1 \iff & \gamma_y = \hat{g}(\vec{\gamma}_x,\vec{\gamma}_w)+\gamma_0\\ & \wedge \mathsf{OpenVer}(\mathsf{ck},\mathsf{com}_\gamma,(\vec{\gamma}_x,\vec{\gamma}_w,\gamma_0),o_\gamma)\end{aligned}$$

In our VC construction we assume the existence of two argument systems $\mathsf{CaP_H}$ and $\mathsf{CaP}_{\hat{\mathsf{g}}}$ for relations $\mathcal{R}_H$ and $\mathcal{R}_{\hat{g}}$ respectively. The construction follows.

$\mathsf{Setup}(\lambda) \to (PK,SK)$:
- Run $(\mathsf{pk},\mathsf{evk},\mathsf{sk}) \leftarrow \mathsf{SHE.KeyGen}(\lambda)$ to generate keys for $\mathsf{SHE}$.
- Run $\mathsf{ck} \leftarrow \mathsf{ComGen}(\lambda)$ to generate a key for a commitment *Com*.
- Set $PK = (\mathsf{pk},\mathsf{evk},\mathsf{ck})$ and $SK = \mathsf{sk}$.

$\mathsf{KeyGen}_{PK}(g) \to (PK_g,SK_g)$:
- Run $\mathsf{crs}_H \leftarrow \mathsf{CaP_H.Setup}(\lambda)$ to generate a common reference string for $\mathcal{R}_H$
- Run $\mathsf{crs}_{\hat{g}} \leftarrow \mathsf{CaP}_{\hat{\mathsf{g}}}.\mathsf{Setup}(\lambda)$ to generate a common reference string for $\mathcal{R}_{\hat{g}}$.
- Prepare a vector $\vec{\omega}_0$ of $z$ honest encryptions of 0.
- Set $PK_g = (PK,\hat{g},\mathsf{crs}_H,\mathsf{crs}_{\hat{g}},\vec{\omega}_0)$ and $SK_g = \emptyset$.

$\mathsf{ProbGen}_{PK}(x) \to (\sigma_x,\tau_x)$:
- Parse $x \in R_t^n$ as $(x_1,x_2,\ldots,x_n)$.

– Run $c_{x,i} \leftarrow \mathsf{SHE.Enc}_{\mathsf{pk}}(x_i)$ for each $i = 1$ to $n$ to get the ciphertexts $c_x = (c_{x,1}, c_{x,2}, \ldots, c_{x,n}) \in \left(R_q^D\right)^n$.
– Run $(\mathsf{com}_x, o_x) \leftarrow \mathsf{Commit}(\mathsf{ck}, c_x)$ to compute the commitment to $c_x$.
– Set $\sigma_x = (c_x, o_x)$ and $\tau_x = \mathsf{com}_x$.

$\langle \mathsf{Compute}_{PK_g}(\sigma_x, w), \mathsf{Verify}_{PK_g}(\tau_x) \rangle$: prover and verifier proceed as follows.
– Compute performs:
  • Parse $w \in R_t^m$ as $(w_1, w_2, \ldots, w_m)$ and let $c_{w,i} = (w_i, 0^{D-1}) \in R_q^D$ for each $i = 1$ to $m$, then get the ciphertext $c_w = (c_{w,1}, c_{w,2}, \ldots, c_{w,m})$.
  • Randomly sample $\vec{\beta} = (\beta_1, \ldots, \beta_z) \xleftarrow{\$} \{0,1\}^z$ and compute $c_0 = \langle \vec{\beta}, \vec{\omega}_0 \rangle$.
  • Compute $c_y = \hat{g}(c_x, c_w) + c_0$ (without reduction modulo $f$).
  • Compute $(\mathsf{com}_w, o_w) \leftarrow \mathsf{Commit}(\mathsf{ck}, c_w)$ and $(\mathsf{com}_0, o_0) \leftarrow \mathsf{Commit}(\mathsf{ck}, c_0)$.
  • Send $(c_y, \mathsf{com}_w, \mathsf{com}_0)$ to Verify.
– Verify samples and sends a homomorphic hash function $H \xleftarrow{\$} \mathcal{H}$ to Compute.
– Verify computes $\gamma_y = H(c_y)$.
– Compute performs:
  • Compute $\vec{\gamma}_x = (H(c_{x,i}))_{i=1}^n, \vec{\gamma}_w = (H(c_{w,i}))_{i=1}^m, \gamma_0 = H(c_0)$.
  • Run $(\mathsf{com}_\gamma, o_\gamma) \leftarrow \mathsf{Commit}((\vec{\gamma}_x, \vec{\gamma}_w, \gamma_0))$;
  • Send the commitment $\mathsf{com}_\gamma$ to Verify.
– Compute and Verify run $\mathsf{CaP}_{\mathsf{H}}$ for the relation $\mathcal{R}_H$ in the roles of prover and verifier respectively. Let $b_1$ be the bit returned by $\mathcal{V}$.
– Compute and Verify run $\mathsf{CaP}_{\hat{\mathsf{g}}}$ for the relation $\mathcal{R}_{\hat{g}}$ in the roles of prover and verifier respectively. Let $b_2$ be the bit returned by $\mathcal{V}$.
– Verify accepts if and only if $b_1 = b_2 = 1$.
– Let $\sigma_y$ include the $c_y$ and the transcript of the interactive argument.

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

In the following theorem we state the security of the construction. The proof is similar to that of Theorem 1.

**Theorem 8.** *Assume that* SHE *is a correct, semantically secure and circuit-private somewhat homomorphic encryption scheme, $\mathcal{H}$ is an $\varepsilon$-universal family of hash functions with $\varepsilon = \mathsf{negl}(\lambda)$, Com is a secure commitment scheme, and* $\mathsf{CaP}_{\mathsf{H}}$ *(resp.* $\mathsf{CaP}_{\hat{\mathsf{g}}}$*) are complete, knowledge-sound and zero-knowledge succinct arguments for the relation $\mathcal{R}_H$ (resp. $\mathcal{R}_{\hat{g}}$). Then the above VC scheme for nondeterministic computation is correct, secure, private, outsourceable, and context-hiding.*

*Proof.* The proof is similar to that of the VC scheme in Section 3.2. The correctness follows from the correctness of SHE and *Com* and the completeness of the argument systems $\mathsf{CaP}_{\mathsf{H}}$ and $\mathsf{CaP}_{\hat{\mathsf{g}}}$; the privacy follows from the semantic security of SHE. The outsourceability follows from the compactness of SHE scheme and the succinctness of the argument systems $\mathsf{CaP}_{\mathsf{H}}$ and $\mathsf{CaP}_{\hat{\mathsf{g}}}$.

To prove the security we define the following games (where $G_i(\mathcal{A}) = b$ denotes that game $G_i$ executed with $\mathcal{A}$ outputs $b$):

**Game $G_0$:** this is the same as $\mathbf{Exp}_{\mathcal{A}}^{Verif}[\mathcal{VC}, \lambda]$ of Section 5.1).

**Game $G_1$:** this is like $G_0$ except that in the first round the adversary is supposed to send $(c_w, o_w)$ and $(c_0, o_0)$, and the game outputs 0 if these are not valid openings of $\mathsf{com}_w$ and $\mathsf{com}_0$ respectively.

By the knowledge-binding of the commitment scheme, we can show that for any adversary $\mathcal{A}$ there is another adversary $\mathcal{A}_1$ such that $\Pr[G_0(\mathcal{A}) = 1] \approx \Pr[G_1(\mathcal{A}_1) = 1]$.

**Game $G_2$:** this is like $G_1$ except that the adversary is also supposed to send $v_H$ and $v_{\hat{g}}$, and the game outputs 0 if these are not valid witnesses for the relations $\mathcal{R}_H$ and $\mathcal{R}_{\hat{g}}$ respectively with the appropriate statements defined during the protocol execution.

By the knowledge-soundness of the argument systems, we can show that for any adversary $\mathcal{A}_1$ there is another adversary $\mathcal{A}_2$ such that $\Pr[G_1(\mathcal{A}_1) = 1] \approx \Pr[G_2(\mathcal{A}_2) = 1]$.

**Game $G_3$:** this is the same as $G_2$ except that the game outputs 0 if there are distinct openings for some of $\mathsf{com}_x, \mathsf{com}_w, \mathsf{com}_0, \mathsf{com}_\gamma$.

By the binding of the commitment it can be shown that for any adversary $\mathcal{A}$ it holds $\Pr[G_2(\mathcal{A}) = 1] \approx \Pr[G_3(\mathcal{A}) = 1]$.

Finally, we claim that $\Pr[G_3(\mathcal{A}) = 1] = \epsilon$. To see this notice that for $G_3$ to output 1 it holds that: $\neg\exists w : g(x, w) = y$ where $y$ is the decryption of $c_y$ (this is the original winning condition); the values $c_w, c_0$ sent in the first round are respectively a valid ciphertext and a valid encryption of 0; $\hat{g}(H(c_x), H(c_w)) + H(c_0) = H(c_y)$. By correctness of homomorphic encryption, $\neg\exists w : g(x, w) = y$ also means that for all valid ciphertexts $c'_w$ and any encryption of 0 $c'_0$ it must be $\hat{g}(c_x, c'_w) + c'_0 \neq c_y$. Hence $\bar{c}_y = \hat{g}(c_x, c'_w) + c'_0 \neq c_y$. However (by the homomorphic property of $H$) $H(\bar{c}_y) = H(c_y)$. Since $c_y$ and $\bar{c}_y$ are set before the sampling of $H$, we have that $\Pr[H(\bar{c}_y) = H(c_y)] = \epsilon$.

The context-hiding follows from the noise flooding on the (output) ciphertext, the zero-knowledge of $\mathsf{CaP}_\mathsf{H}$ and $\mathsf{CaP}_{\hat{\mathsf{g}}}$, and the hiding property of the commitment scheme.

Formally, we use Lemma 5 and show how to construct the simulators $S_\tau, S_1, S_2$, and $S_{3,b}$. $S_1$ runs exactly the same as $\mathsf{Setup}$ with $td = \emptyset$, and $S_2$ proceeds as $\mathsf{KeyGen}_{PK}$ except that it also sets $td_g = (td_{\mathsf{CaP}_\mathsf{H}}, td_{\mathsf{CaP}_{\hat{\mathsf{g}}}})$ where $td_{\mathsf{CaP}} = \mathsf{trap} \leftarrow \mathsf{CaP}_\mathsf{H}.\mathsf{Sim}$ and $td_{\mathsf{CaP}_{\hat{\mathsf{g}}}} = \mathsf{trap} \leftarrow \mathsf{CaP}_{\hat{\mathsf{g}}}.\mathsf{Sim}$. $S_\tau$ can be the simulator that outputs a commitment to a dummy message, which is indistinguishable from $\tau_x$ by the hiding of the commitment.

Finally, $S_{3,b}(\tau_x, y)$ sets $c_y$ as an $\mathsf{SHE}$ encryption of $y$ built by using the circuit privacy simulator, and computes $\mathsf{com}_w, \mathsf{com}_0$ as commitments to dummy messages; next it runs the zero-knowledge simulators of $\mathsf{CaP}_\mathsf{H}$ and $\mathsf{CaP}_{\hat{\mathsf{g}}}$. □

*Remark 13.* We observe that the check $c_w \in \Omega^m$ can express conditions that are more specialized than the one we mention above (i.e., that it is a valid encoding of a plaintext). This can be useful to enforce some checks on the non-deterministic input $w$ which may not be expressed with a constant-degree polynomial, and can be pushed to the relation $\mathcal{R}_H$ and the proof system $\mathsf{CaP}_\mathsf{H}$.

**On the instantiation of $\mathsf{CaP}_\mathsf{H}$ and $\mathsf{CaP}_{\hat{\mathsf{g}}}$.** We observe that $\mathsf{CaP}_{\hat{\mathsf{g}}}$ can be instantiated with any succinct commit-and-prove ZK argument for proving the correctness of arithmetic circuits over $\mathbb{Z}_q[X]/(h)$ (which can also be efficiently instantiated via one for $\mathbb{Z}_q$). $\mathcal{R}_H$ instead is best modeled as a circuit over $\mathbb{Z}_q$, where: for each hashing one checks a Euclidean division (with evaluation in $R$ for double hashing); $\beta_i \in \{0, 1\}$ can be encoded with the classical equation $\beta_i^2 - \beta_i = 0$ (which works for any $q$ that is a prime power); and the check $c_{w,i} \in \Omega$ requires $d_f$ range proofs in an interval of size $t$ and $d_f(D - 1)$ equal-to-0 checks.

To the best of our knowledge, there are no commit-and-prove SNARKs that can directly handle circuits over Galois rings such as $\mathbb{Z}_q[X]/(h)$. One could however rely on one for circuits over $\mathbb{Z}_q$. When $q$ is prime, there are many already existing choices, such as the ones from [FNP20] for $q \geq 2^\lambda$, or Ligero [AHIV17] which can work for any prime $q$ and can be repeated to increase its soundness if $q$ is too small and offers an efficiency gain with batched proofs.

### 5.5 A Variant Context-Hiding VC with Public Hashed Ciphertexts

In this section, we present a generic scheme achieving the same goal — proving nondeterministic computation with context-hiding — as the previous one (in Section 5.4) but exploiting only *one* $\mathsf{CaP}$ argument instead of *two*. The scheme leverages the fact that the image of hash functions on an input does *not* leak the information on the input given that the number of images is not large. With this, it uses the $\mathsf{CaP}$ argument (similarly to $\mathsf{CaP}_{\mathsf{H}}$) only for the evaluation of hash, discloses the hashed output to $\mathcal{V}$, then uses an argument, instead of a $\mathsf{CaP}$ argument, for proving the computation on them.

We first define the *hiding* property of a hash family.

**Definition 12 (Hiding).** *Let $\mathcal{H}$ be a family of hash functions $H : \mathbb{Z}_q[X]^D \to \mathcal{D}_H$ and $\mathcal{D}$ be a distribution on a subset of $\mathbb{Z}_q[X]^D$. We say that $\mathcal{H}$ is $\delta_s$-hiding on $\mathcal{D}$ for $N$-queries if, for $c \xleftarrow{\$} \mathcal{D}$, $H_i \xleftarrow{\$} \mathcal{H}$, and $u_i \xleftarrow{\$} \mathcal{D}_H$, it holds that*

$$(H_i(c), H_i)_{i=1}^N \overset{stat}{\approx} (u_i, H_i)_{i=1}^N$$

*where the statistical distance between the above distributions is less than $\delta_s$, and we assume that $H_i$'s are all different from each other.*

We will show, in the following section, that our homomorphic hash family (from Section 4.3) satisfies this *hiding* property.

To present our VC scheme with hiding homomorphic hash family, we first introduce the NP relations and corresponding commit-and-prove argument ($\mathsf{CaP}$) and an argument system that will be required in the VC scheme. We use the same notation as Section 5.4 for the computation $g$, $\hat{g}$, and the commitment scheme $Com$ with a key $\mathsf{ck}$.

A $\mathsf{CaP}$ argument $\mathsf{CaP}'_{\mathsf{H}}$ proves the evaluation of hash functions, i.e., the following relation $\mathcal{R}'_H$ on $u'_H = (\vec{\omega}_0, H, (\vec{\gamma}_x, \vec{\gamma}_w, \gamma_0), \mathsf{com}_x, \mathsf{com}_w, \mathsf{com}_0)$ and $v'_H = (c_x, c_w, c_0, o_x, o_w, o_0, \vec{\beta})$:

$$\begin{aligned}\mathcal{R}'_H(u'_H, v'_H) = 1 \iff & \vec{\gamma}_x = H(c_x) \wedge \vec{\gamma}_w = H(c_w) \wedge \gamma_0 = H(c_0) \\ & \wedge c_0 = \langle \vec{\beta}, \vec{\omega}_0 \rangle \wedge \vec{\beta} \in \{0,1\}^z \\ & \wedge c_w \in \Omega^m \\ & \wedge \mathsf{OpenVer}(\mathsf{ck}, \mathsf{com}_x, (c_{x,1}, \ldots, c_{x,n}), o_x) \\ & \wedge \mathsf{OpenVer}(\mathsf{ck}, \mathsf{com}_w, (c_{w,1}, \ldots, c_{w,m}), o_w) \\ & \wedge \mathsf{OpenVer}(\mathsf{ck}, \mathsf{com}_0, c_0, o_0)\end{aligned}$$

where $\vec{\omega}_0$ is a vector of encryptions of 0, and $c_w \in \Omega^m$ is a shorthand for checking that each $c_{w,i}$ is a valid encryption. Note that the two differences of $\mathcal{R}'_H$ from $\mathcal{R}_H$ are that $(\vec{\gamma}_x, \vec{\gamma}_w, \gamma_0)$ is disclosed as a part of public input $u'_H$ and that the $c_w$ is generated as fresh ciphertexts.[21]

On the other hand, an argument $\Pi_{\hat{\mathsf{g}}}$ proves the computation on the hashed outputs, i.e., the following relation $\mathcal{R}'_{\hat{g}}$ on $u'_{\hat{g}} = (\gamma_y, \vec{\gamma}_x, \vec{\gamma}_w, \gamma_0)$:

$$\mathcal{R}'_{\hat{g}}(u'_{\hat{g}}) = 1 \iff \gamma_y = \hat{g}(\vec{\gamma}_x, \vec{\gamma}_w) + \gamma_0.$$

[21] We mention that this generation procedure can be a relaxed one than the original encryption algorithm, e.g., without error sampling which is costly to prove, since only the image of $c_w$ under a hash will be disclosed, i.e., the output $c_w$ does not need to hide $w$ but only needs to contain enough randomnesses so that its image under a hash does not reveal much information of it (see Lemma 6 and the following Remark).

Now, the construction of our VC follows with the above two argument systems $\mathsf{CaP}'_{\mathsf{H}}$ and $\Pi'_{\hat{\mathsf{g}}}$ for the relations $\mathcal{R}'_H$ and $\mathcal{R}'_{\hat{g}}$, respectively. Most part of the scheme is the same as the previous one (in Section 5.4), but the main difference is that the prover in $\mathsf{Compute}$ sends the output $(\vec{\gamma}_x, \vec{\gamma}_w, \gamma_0)$ of hash $H$ directly, instead of the commitments, to the verifier.

$\mathsf{Setup}(\lambda) \to (PK, SK)$:
- The same as the previous scheme: generate keys for the building blocks.

$\mathsf{KeyGen}_{PK}(g) \to (PK_g, SK_g)$:
- The same as the previous scheme, except that $\mathsf{crs}'_H \leftarrow \mathsf{CaP}'_{\mathsf{H}}.\mathsf{Setup}(\lambda)$ (for $\mathcal{R}'_H$) and $\mathsf{crs}'_{\hat{g}} \leftarrow \Pi_{\hat{\mathsf{g}}}.\mathsf{Setup}(\lambda)$ (for $\mathcal{R}'_{\hat{g}}$).
- Set $PK_g = (PK, \hat{g}, \mathsf{crs}'_H, \mathsf{crs}'_{\hat{g}}, \vec{\omega}_0)$ and $SK_g = \emptyset$.

$\mathsf{ProbGen}_{PK}(x) \to (\sigma_x, \tau_x)$:
- The same as the previous scheme: set $\sigma_x = (c_x, o_x)$ and $\tau_x = \mathsf{com}_x$ where $c_x = (\mathsf{SHE.Enc}_{\mathsf{pk}}(x_i))_i$.

$\langle \mathsf{Compute}_{PK_g}(\sigma_x), \mathsf{Verify}_{PK_g}(\tau_x) \rangle$: prover and verifier proceed as follows.
- $\mathsf{Compute}$ performs:
  - The same as the previous scheme except that $c_w$ is generated with enough entropy (see footnote 21): Sends $(c_y, \mathsf{com}_w, \mathsf{com}_0)$ to $\mathsf{Verify}$.
- $\mathsf{Verify}$ samples and sends a homomorphic hash function $H \xleftarrow{\$} \mathcal{H}$ to $\mathsf{Compute}$.
- $\mathsf{Verify}$ computes $\gamma_y = H(c_y)$.
- $\mathsf{Compute}$ performs:
  - Compute $\vec{\gamma}_x = (H(c_{x,i}))_{i=1}^n, \vec{\gamma}_w = (H(c_{w,i}))_{i=1}^m, \gamma_0 = H(c_0)$.
  - Send $(\vec{\gamma}_x, \vec{\gamma}_w, \gamma_0)$ to $\mathsf{Verify}$.
- $\mathsf{Compute}$ and $\mathsf{Verify}$ run $\mathsf{CaP}'_{\mathsf{H}}$ for the relation $\mathcal{R}'_H$ in the roles of prover and verifier respectively. Let $b_1$ be the bit returned by $\mathcal{V}$.
- $\mathsf{Compute}$ and $\mathsf{Verify}$ run $\Pi'_{\hat{\mathsf{g}}}$ for the relation $\mathcal{R}'_C$ in the roles of prover and verifier respectively. Let $b_2$ be the bit returned by $\mathcal{V}$.
- $\mathsf{Verify}$ accepts if and only if $b_1 = b_2 = 1$.
- Let $\sigma_y$ include $c_y$ and the transcript of the interactive argument.

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

Note that the argument system $\Pi'_{\hat{\mathsf{g}}}$ is more efficient and easy to instantiate than the $\mathsf{CaP}$ argument $\mathsf{CaP}_{\hat{\mathsf{g}}}$ of the previous scheme since it does not deal with the proof of opening the commitment. The construction also satisfies the required properties as follows.

**Theorem 9.** *Assume that* $\mathsf{SHE}$ *is a correct, semantically secure and circuit-private somewhat homomorphic encryption scheme,* $\mathcal{H}$ *is an* $\varepsilon$*-universal family of hash functions with* $\varepsilon = \mathsf{negl}(\lambda)$ *and is* $\delta_s$*-hiding on the distribution of encryption of fresh ciphertext (having the input* $x$ *or the witness* $w$ *as messages) for* $N$*-queries, Com is a secure commitment scheme, and* $\mathsf{CaP}'_{\mathsf{H}}$ *is a complete, knowledge-sound and zero-knowledge succinct argument for the relation* $\mathcal{R}_H$ *while* $\Pi'_{\hat{\mathsf{g}}}$ *is a complete, sound, and succinct argument for the relation* $\mathcal{R}_{\hat{g}}$*. Then the above VC scheme for nondeterministic computation is correct, secure, private, outsourceable, and context-hiding, given that there has been at most* $N$ *distinct proofs (with different hashes) on the same input.*

*Proof.* Other properties than the context-hiding follows similarly as the previous proof of Theorem 8. The context-hiding also follows similarly using Lemma 5 and showing the existence of simulators $S_\tau, S_1, S_2$, and $S_{3,b}$ as follows: $S_1$ runs exactly the same as $\mathsf{Setup}$ with $td = \emptyset$, and $S_2$ proceeds

as $\mathsf{KeyGen}_{PK}$ except that it also sets $td_g = \mathsf{trap} \leftarrow \mathsf{CaP}'_{\mathsf{H}}.\mathsf{Sim}$. The hiding of commitment scheme guarantees the existence of a simulator $S_\tau$ outputting commitment to dummy messages, which is indistinguishable from $\tau_x$. Finally, $S_{3,b}$ sets $\sigma_y^*$ as composed of an $\mathsf{SHE}$ encryption $c_y^*$ of $g(x,w)$ with appropriate noises, the proof $\pi^*_{\mathsf{CaP}'_{\mathsf{H}}} \leftarrow \mathsf{CaP}'_{\mathsf{H}}.\mathsf{Sim}(td_g, u'^*_H)$, and the transcripts of $\Pi'_{\hat{\mathsf{g}}}$ on $u'^*_{\hat{g}}$ where $u'^*_H$ and $u'^*_{\hat{g}}$ are the same as $u'_H$ and $u'_{\hat{g}}$ except that the commitments other than $\mathsf{com}_x$ are dummy commitments and that $\gamma_y = H(c_y^*)$ and $(\vec{\gamma}_x, \vec{\gamma}_w, \gamma_0)$ are substituted by $(\vec{\gamma} \xleftarrow{\$} \mathcal{D}_H^{n+m}, H(c_y^*) - \hat{g}(\vec{\gamma}))$.

Then, the indistinguishability of simulated output and real output follows from the hybrid argument with consecutive hybrid simulators $S_{3,b}^{hyb-0}$, $S_{3,b}^{hyb-1}$, $S_{3,b}^{hyb-2}$, $S_{3,b}^{hyb-3}$ as follows:

- $S_{3,b}^{hyb-0}$: the real execution.
- $S_{3,b}^{hyb-1}$: the same as $S_{3,b}^{hyb-0}$ except that it outputs $c_y^*$ and $H(c_y^*) - \hat{g}(\vec{\gamma}_x, \vec{\gamma}_w)$ instead of $c_y$ and $\gamma_0$, respectively.
- $S_{3,b}^{hyb-2}$: the same as $S_{3,b}^{hyb-1}$ except that it outputs a dummy commitment.
- $S_{3,b}^{hyb-3}$: the $S_{3,b}$.

Note that the indistinguishability of $S_{3,b}^{hyb-0}$ and $S_{3,b}^{hyb-1}$ follows from that of $c_y$ and $c_y^*$, i.e., the circuit privacy of $\mathsf{SHE}$; the indistinguishability of $S_{3,b}^{hyb-1}$ and $S_{3,b}^{hyb-2}$ (resp., $S_{3,b}^{hyb-2}$ and $S_{3,b}^{hyb-3}$) follows from the hiding property of commitment scheme (resp., from the hiding property of hash family). □

**Hiding Property of Our Hash Function Family**.

We now show that the homomorphic hash function family, especially, the double hash (Definition 8) introduced in Section 4.3 satisfies the *hiding* property (Definition 12) described above. Recall that the fresh $\mathsf{BV}$ ciphertext is of the form $c_0 + c_1 Y$ with $c_0 = -c_1 s + te + m$ and the double hash is defined by $H_{r,h}(c_0 + c_1 Y) := c_0 + c_1 r \bmod h$.

**Lemma 6.** *Let $\mathcal{H} := \{H_{r,h}\}$ be the family of homomorphic hash functions (Definition 8) where $h \in \mathbb{Z}_q[X]$ 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 6). Let $\mathcal{D}_m$ be the distribution of fresh ciphertexts ($\subset \mathbb{Z}_q[X]^2 \cong \mathbb{Z}_q[X][Y]$) from the encryption algorithm $\mathsf{BV.Enc_{pk}}(m)$ for a message $m \in \mathbb{Z}_t[X]/(f)$. Assume that $N \le \frac{d_f}{d_h}$, then for all message $m$, $\mathcal{H}$ is $\delta_s$-hiding on $\mathcal{D}_m$ for $N$-queries with $\delta_s = \frac{N}{p^{d_h}}$.*

*In other words, for all $aY + b \xleftarrow{\$} \mathcal{D}_m$, it holds that*

$$((H_{r_i,h_i}(aY+b), r_i, h_i))_{i=1}^N \overset{stat}{\approx} ((u_i, r_i, h_i))_{i=1}^N$$

*where $u_i \xleftarrow{\$} \mathbb{Z}_q[X]/(h_i)$ and $\{r_i, h_i\}_{i=1}^N$ are all different from each other, and the statistical distance is bounded by $\frac{N}{p^{d_h}}$.*

*Proof.* Since $\{h_i\}_{i=1}^N$ are distinct monic irreducible polynomials, they are coprime to each other, and we get the ring isomorphism $\mathbb{Z}_q[X]/(h_1 h_2 ... h_N) \simeq \prod_{i=1}^N \mathbb{Z}_q[X]/(h_i)$ by Chinese remainder theorem. Let $a_i$ be the image of projection of $a$ on $\mathbb{Z}_q[X]/(h_i)$ from the isomorphism. Since $(a,b)$ is a RLWE sample, $a \xleftarrow{\$} \mathbb{Z}_q^{d_f} \cong \mathbb{Z}_q[X]/(f)$ and each $a_i$ is uniformly random in $\mathbb{Z}_q[X]/(h_i)$. Therefore, if $r_i$ is not a zero divisor, $(a_i r_i, r_i) \approx (u_i, r_i)$ since the map $x \to r_i x$ is one-to-one in $\mathbb{Z}_q[X]/(h_i)$. Hence,

$$\{(a_i r_i, r_i, h_i)\}_{i=1}^N \overset{\text{stat}}{\approx} \{(u_i, r_i, h_i)\}_{i=1}^N$$

where the statistical distance is bounded by the probability that at least one of $r_i$ is a zero divisor, which is $\frac{N}{p^{d_h}}$ (since $r_i \xleftarrow{\$}$ the sampling set $A$ (Definition 6) is a zero divisor with probability $\frac{1}{p^{d_h}}$). Now, the theorem follows from the definition of $H_{r_i,h_i}$, i.e. from the fact that $H_{r_i,h_i}(aY+b) = a_i r_i + b_i$ where $b_i$ is the image of projection of $b$ on $\mathbb{Z}_q[X]/(h_i)$.

□

*Remark 14.* The indistinguishability of above lemma indeed originates from the random sampling of $a$ and holds also in the presence of the secret key of BV scheme. Recall that one should set $p^{d_h} \approx 2^\lambda$ to achieve $\epsilon = \mathsf{negl}(\lambda)$-universality (Theorem 4) of the hash which is necessary for the soundness of the VC scheme. Therefore, given that $N \leq \frac{d_f}{d_h}$, we can expect that the hash is $\delta_s$-hiding with $\delta_s = \mathsf{negl}(\lambda)$ when exploited in the VC scheme.

*Remark 15.* After reaching the maximal number of allowed queries, $\mathcal{P}$ can re-randomize the ciphertexts to answer another queries on the ciphertexts. The re-randomization, for example, can be done by adding encryptions of zero to the ciphertexts (similarly as the noise flooding technique) then publishing commitments of new ciphertexts with the appropriate proof (of commit-and-prove argument) that $\mathcal{P}$ did the re-randomization correctly.

We finally mention that a family of single hash functions ($H_h(a) = a \bmod h \in \mathbb{Z}_q[X]/(h)$, Definition 7) does not seem to satisfy the *hiding* property, especially when the secret key $s$ is known, since the image $(H_h(a), H_h(a)H_h(s) + H_h(e))$ of a RLWE sample $(a, as + e)$ can be close to the uniform distribution only if $H_h(e)$ is close to the uniform distribution.


### 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), and SECURITAS (ref. RED2018-102321-T), by the Madrid Regional Government under project BLOQUES (ref. S2018/TCS-4339), and by a research grant from Nomadic Labs and the Tezos Foundation.


## 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.

Alb17. M. R. Albrecht. On Dual Lattice Attacks Against Small-Secret LWE and Parameter Choices in HElib and SEAL. In J.-S. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 103–129. Springer, Heidelberg, April / May 2017.

APS15. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

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.

BEHZ16. J.-C. Bajard, J. Eynard, M. A. Hasan, and V. Zucca. A full RNS variant of FV like somewhat homomorphic encryption schemes. In *International Conference on Selected Areas in Cryptography*, pages 423–442. Springer, 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.

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.

CHK+18. J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song. A full RNS variant of approximate homomorphic encryption. In *International Conference on Selected Areas in Cryptography*, pages 347–368. Springer, 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.

HPS19. S. Halevi, Y. Polyakov, and V. Shoup. An Improved RNS Variant of the BFV Homomorphic Encryption Scheme. In M. Matsui, editor, *CT-RSA 2019*, volume 11405 of *LNCS*, pages 83–105. Springer, Heidelberg, March 2019.

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.

LFKN92. C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM (JACM)*, 39(4):859–868, 1992.

LP11. R. Lindner and C. Peikert. Better Key Sizes (and Attacks) for LWE-Based Encryption. In A. Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Heidelberg, February 2011.

LPR10. V. Lyubashevsky, C. Peikert, and O. Regev. On Ideal Lattices and Learning with Errors over Rings. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May / June 2010.

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.

Rot09. G. N. Rothblum. *Delegating computation reliably: paradigms and constructions*. PhD thesis, Massachusetts Institute of Technology, 2009.

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.

## A BV Homomorphic Encryption Scheme

Let $\chi$ denote a discrete Gaussian distribution over the ring $R := \mathbb{Z}[X]/(f)$ (identified as a set $(\mathbb{Z})^{d_f}$) with standard deviation $\sigma$. We identify $R_q := \mathbb{Z}_q[X]/(f)$ as a subset $(\mathbb{Z} \cap [-\frac{q}{2}, \frac{q}{2}))[X]/(f)$ of $R$ when required (and similarly for $R_t$). The formal description of BV $\mathsf{SHE}$ scheme from [BV11] is as follows:

$\mathsf{BV.KeyGen}(\lambda) \to (\mathsf{pk}, \mathsf{dk})$**:** sample a secret key $\mathsf{dk} = s \xleftarrow{\$} \chi$, and a public key $\mathsf{pk} = (a_0, b_0 = a_0 s + te_0)$ where $a_0 \xleftarrow{\$} R_q$ and $e_0 \xleftarrow{\$} \chi$.

$\mathsf{BV.Enc}_{\mathsf{pk}}(m) \to c$**:** sample $v, e' \xleftarrow{\$} \chi$ and $e'' \xleftarrow{\$} \chi'$, where $\chi'$ is the same as $\chi$ but with standard deviation $\sigma' > 2^{\omega(\log\, d_f)} \cdot \sigma$. Then compute $(a, b) = (a_0 v + te', b_0 v + te'') \in R_q^2$. Output $c = (c_0, c_1, 0, ..., 0) \in R_q^D$ where:

$$c_0 = m + b \ , \ c_1 = -a$$

$\mathsf{BV.Dec}_{\mathsf{dk}}(c) \to m$**:** using the secret key $\mathsf{dk} = s$, compute $c(s) = \sum_{i=0}^{D-1} c_i s^i$ and output $m = c(s) \ mod \ t$.

$\mathsf{BV.Eval}(g, c_1, \ldots, c_t) \to c_g$**:** addition and multiplication of two ciphertexts $c = \mathsf{Enc}(m), c' = \mathsf{Enc}(m') \in R_q^D$ are defined as the usual addition and multiplication in $R_q[Y]$:

- $c + c' = (c_0 + c'_0, .., c_{D-1} + c'_{D-1})$ encrypts $m + m'$.
- we see the ciphertexts as polynomials as explained before. Let $c(Y) \cdot c'(Y) = c^* = \sum_{i \geq 0} c_i^* Y^i$: Then $m \cdot m'$ is encrypted by $(c_0^*, ..., c_{D-1}^*)$.

Note that in the case of multiplication, the result will not be correct if the degree of $c^*$ (in $Y$) exceeds $D - 1$.

The condition for the correctness and the security are as follows.

**Lemma 7 (Correctness and Security [BV11]).** *Let $\sigma$ be the standard deviation of the noise distribution (used in* $\mathsf{Enc}$*) in* $\mathsf{BV}$ *scheme. Then,* $\mathsf{BV}$ *scheme is semantically secure under the* $\mathsf{PLWE}_{f,q,\chi}$ *assumption, and it correctly computes any multi-variate polynomial $g$ of degree $D-1$ over $R_t$ given that*

$$||g||_\infty (t \cdot \sigma \cdot d_f^{1.5})^{(D-1)} \leq \frac{q}{2},$$

*where $||g||_\infty$ denotes the size of maximal coefficient of $g$, and other values are as described in Section 4.1,* **Parameters**.

## B The GKR Protocol over Rings

For completeness, we present the GKR protocol ([GKR08, Tha13]), which is a (public-coin) interactive proof system, whose generalized version for arithmetic circuits over Galois rings was exploited in our instantiation of the VC scheme.

### B.1 Preliminaries

**Schwartz-Zippel Lemma**. We start with the famous lemma which plays a central role in the GKR protocol.

**Lemma 8.** *Let $\mathbb{F}$ be a field, let $f$ be an $n$-variate polynomial of total degree $d_f$. Then for any finite subset $A \subseteq \mathbb{F}$ such that $d_f \leq |A|$,*

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

**Sum-check Protocol**. We present the famous Sum-check protocol which will be called as a subroutine in the GKR protocol.

**Theorem 10 (Sum-check Protocol [LFKN92]).** *Let $\mathbb{F}$ be a finite field. For an $n$-variate polynomial $f : \mathbb{F}^n \to \mathbb{F}$ of degree at most $d \leq |\mathbb{F}|$ in each variable and $\beta \in \mathbb{F}$, the sum-check protocol is an interactive proof protocol to prove that $S(f) = \beta$, where*

$$S(f) := \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} f(b_1, b_2, \ldots, b_n).$$

*The protocol has the soundness probability $\frac{nd}{|\mathbb{F}|}$. The computational complexity, measured by the number of operations over $\mathbb{F}$ required, is $O(2^n d)$ for the prover $\mathcal{P}$ and $O(nd)$ for the verifier $\mathcal{V}$. The communication cost (the number of $\mathbb{F}$-elements transffered) is $O(nd)$, and the space complexity of $\mathcal{V}$ is $O(d + n)$.*

*Proof.* The proof of soundness can be found in [LFKN92] while the derivation of cost can also be found in [Tha13]. In a nutshell, the soundness is guaranteed from the Schwarz-Zippel Lemma (Lemma 8): (see the following description of the sum-check protocol) if a prover have sent an incorrect value $\beta'$, then he must send, in the first round, a polynomial $g_1$ which is different from $f_1$. Then, $g_1(r_1) \neq f_1(r_1)$ ($r_1 \xleftarrow{\$} \mathbb{F}$) with high probability and it forces the prover to send a polynomial $g_2$ which is different from $f_2$. Continuing this process, the prover has less than $\frac{nd}{|\mathbb{F}|}$ probability of passing all checks of verifier during total $n$ rounds. □

We give a descriptions of the Sum-check protocol:

- The protocol proceeds in $n$ rounds.
- In the first round, $\mathcal{P}$ sends

$$f_1(t) := \sum_{b_2 \in \{0,1\}} \sum_{b_3 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} f(t, b_2, b_3, \ldots, b_n),$$

  and $\mathcal{V}$ checks $\beta = f_1(0) + f_1(1)$. Then, $\mathcal{V}$ sends $r_1 \xleftarrow{\$} \mathbb{F}$ to $\mathcal{P}$.
- At the $i$-th round ($2 \leq i \leq n-1$), $\mathcal{P}$ sends

$$f_i(t) := \sum_{b_{i+1} \in \{0,1\}} \sum_{b_{i+2} \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} f(r_1, \ldots, r_{i-1}, t, b_{i+1}, \ldots, b_n),$$

  and $\mathcal{V}$ checks[22] $f_{i-1}(r_{i-1}) = f_i(0) + f_i(1)$. Then, $\mathcal{V}$ sends $r_i \xleftarrow{\$} \mathbb{F}$ to $\mathcal{P}$.

[22] It also checks that $f_i(t)$ is a polynomial of degree at most $d$.

– At the final round, $\mathcal{P}$ sends
$$f_n(t) := f(r_1, r_2, \ldots, r_{n-1}, t),$$
and $\mathcal{V}$ checks $f_{n-1}(r_{n-1}) = f_n(0)+f_n(1)$. Then, $\mathcal{V}$ checks, with $r_n \xleftarrow{\$} \mathbb{F}$, if $f_n(r_n) = f(r_1, r_2, \ldots, r_n)$.
– If all checks pass, $\mathcal{V}$ accepts. Otherwise, rejects.

**Multilinear Extension**. We finally recall the multilinear extension.

**Lemma 9 (Multilinear Extension (MLE) [CMT12]).** *Given a function $V : \{0,1\}^n \to \mathbb{F}$, there exists a unique multilinear (i.e., linear in each variable, e.g., $f(x_1, x_2) = ax_1x_2 + bx_1 + cx_2$) polynomial $\tilde{V}(\vec{x}) : \mathbb{F}^n \to \mathbb{F}$ extending $V$, i.e., $\tilde{V}(\vec{x}) = V(\vec{x})$ for all $\vec{x} \in \{0,1\}^n$. We call $\tilde{V}$ the* multilinear extension *of $V$ over $\mathbb{F}$.*

Explicitly, the multilinear extension of $V$ can be defined as follows from which the above lemma follow.
$$\tilde{V}(x_1, x_2, \ldots, x_n) = \sum_{\vec{b} \in \{0,1\}^n} V(\vec{b}) \cdot \prod_{i=1}^{n} [(1 - b_i)(1 - x_i) + b_i x_i].$$

### B.2 The GKR Protocol

**Theorem 11 (The GKR protocol [GKR08, CMT12, Tha13, XZZ+19]).** *Let $C : \mathbb{F}^n \to \mathbb{F}$ be a (layerd) circuit over a finite field $\mathbb{F}$ with fan-in 2, of size $S$ and depth $d$. The GKR protocol is an interactive proof system that allows a prover $\mathcal{P}$ to prove that $C(x) = y$ to a verifier $\mathcal{V}$ with the following properties:*

**Completeness:** *if $C(x) = y$, $\Pr(\mathcal{V}$ accepts$) = 1$*
**Soundness:** *if $C(x) \neq y$, $\Pr(\mathcal{V}$ accepts$) \leq \frac{7d\log S}{|\mathbb{F}|}$*
**Complexity:** *$O(S)$ for $\mathcal{P}$, $O(n + d \log S)$[23] for $\mathcal{V}$ (with $O(\log S)$ space), and $O(d \log S)$ for the communication cost.*

*Proof.* The proof of soundness can be found in [GKR08, Rot09], and the derivation of cost can be found in one of the latest work, e.g., [XZZ+19]. Briefly, the correctness and soundness of the GKR protocol follows from those of the Sum-Check protocol and the Schwartz-Zippel lemma. □

We give an overview and a description of the GKR protocol.
**Notation**. We number the layers of given circuit $C$ consecutively in a way that the output layer is 0-th layer and the input layer is $d$-th layer. For simplicity, we assume that the size $S_i$ of $i$-th layer as $2^{s_i}$, a power-of-two, and number each gate of the $i$-th layer as one of $j \in \{0,1\}^{s_i}$. Then, the functions $V_i : \{0,1\}^{s_i} \to \mathbb{F}$ are defined in a way that $V_i(j)$ is the output of the $j$-th gate at the $i$-th layer. Let $\tilde{V}_i : \mathbb{F}^{s_i} \to \mathbb{F}$ denote the multilinear extension of $V_i$.

For each $i \in [0, d) \cap \mathbb{Z}$, we also define functions $add_i$ and $mult_i$ both mapping $\{0,1\}^{s_i} \times \{0,1\}^{s_{i+1}} \times \{0,1\}^{s_{i+1}} \to \{0,1\}$ that specify the wiring pattern of the circuit $C$:

Let $g_{i,j}$ denotes the output of $j$-th gate at $i$-th layer. Then,
$$add_i(z, w_1, w_2) := \begin{cases} 1 \text{ if } g_{i,z} = g_{i+1,w_1} + g_{i+1,w_2} \\ 0 \text{ otherwise} \end{cases}$$

[23] Given that the wiring predicate of circuit efficiently computable in $O(\log S)$ complexity.

$$mult_i(z, w_1, w_2) := \begin{cases} 1 \text{ if } g_{i,z} = g_{i+1,w_1} \times g_{i+1,w_2} \\ 0 \text{ otherwise} \end{cases}$$

In other words, $add_i(z, w_1, w_2) = 1$ only if the $z$-th gate of $i$-th layer is an addition gate taking the output of $w_1$-th gate and $w_2$-th gate at $i+1$-th layer, and $mult_i$ is defined similarly for multiplication gates. Let $\widetilde{add}_i(z, w_1, w_2)$ (and $\widetilde{mult}_i(z, w_1, w_2)$) be the multilinear extension of $add_i$ (resp. $mult_i$).

**Overview and Protocol Description**. The protocol proceeds in layer by layer from the output layer to the input layer: a verifier, given the claimed output from a prover, reduces (via the sum-check protocol) the claim on the output to the claim on the output of the previous layer; iterating this procedure from the $i$-th layer to the $i+1$-th layer for each $i$, the verifier finally gets the claim on the output of the $d$-th layer, which is indeed the input of the circuit; then, the claim can be checked by the verifier itself with the input.

Now, we describe the detailed procedure. We omit the vector notation.

**The 0-th (Output) layer:** a verifier $\mathcal{V}$, given the claimed output (i.e., the output of gates in the 0-th layer) from a prover $\mathcal{P}$, evaluates the multilinear extension $\tilde{V}_0$ at a random vector $r_0 \in \mathbb{F}^{s_0}$ to get the claim $\tilde{V}_0(r_0) = v_0$ ($\mathcal{V}$ sends $r_0$, $v_0$ to $\mathcal{P}$).

**From $i$-th layer to $i+1$-th layer:** $(0 \le i \le d-1)$

- $\mathcal{V}$ having the claim $\tilde{V}_i(r_i) = v_i$ executes the sum-check protocol, with $\mathcal{P}$ as a prover, on the following equation with $z = r_i$:

$$\tilde{V}_i(z) = \sum_{w_1, w_2 \in \{0,1\}^{s_{i+1}}} [\widetilde{add}_i(z, w_1, w_2)(\tilde{V}_{i+1}(w_1) + \tilde{V}_{i+1}(w_2)) + \widetilde{mult}_i(z, w_1, w_2)(\tilde{V}_{i+1}(w_1)\tilde{V}_{i+1}(w_2))]$$

  Note, at the end of the final round of sum-check, that $\mathcal{V}$ needs to check the following equation (where $r_{w_1}, r_{w_2}$ are sampled by $\mathcal{V}$) for some $A_{i+1}$ given from the sum-check protocol:

$$\widetilde{add}_i(r_i, r_{w_1}, r_{w_2})(\tilde{V}_{i+1}(r_{w_1}) + \tilde{V}_{i+1}(r_{w_2})) + \widetilde{mult}_i(z, r_{w_1}, r_{w_2})(\tilde{V}_{i+1}(r_{w_1})\tilde{V}_{i+1}(r_{w_2})) = A_{i+1}. \tag{7}$$

- $\mathcal{P}$ takes the line $\gamma : \mathbb{F} \to \mathbb{F}^{s_{i+1}}$ such that $\gamma(0) = r_{w_1}$ and $\gamma(1) = r_{w_2}$, then sends $\tilde{V}_{i+1}(\gamma(t)) : \mathbb{F} \to \mathbb{F}$ (which is a univariate polynomial of degree at most $s_{i+1}$) to $\mathcal{V}$.
- $\mathcal{V}$, given $\tilde{V}_{i+1}(\gamma(t))$ from $\mathcal{P}$, gets $\tilde{V}_{i+1}(r_{w_1}) = \tilde{V}_{i+1}(\gamma(0))$ and $\tilde{V}_{i+1}(r_{w_2}) = \tilde{V}_{i+1}(\gamma(1))$, computes $\widetilde{add}_i(r_i, r_{w_1}, r_{w_2})$ and $\widetilde{mult}_i(r_i, r_{w_1}, r_{w_2})$ by itself, then checks if the above equation (7) holds.
- If the check passes, $\mathcal{V}$ samples a random $s \in \mathbb{F}$ and gets the claim $\tilde{V}_{i+1}(r_{i+1}) = \tilde{V}_{i+1}(\gamma(s)) = v_{i+1}$ where $r_{i+1} := \gamma(s)$.
- $\mathcal{V}$ sends $r_{i+1}, v_{i+1}$ to $\mathcal{P}$ and iterates above process with the claim (on $\tilde{V}_{i+1}$) until $i+1 \le d$.

**The $d$-th (Input) layer:** after the above process with $i = d-1$, $\mathcal{V}$ gets the claim $\tilde{V}_d(r_d) = v_d$. Then, $\mathcal{V}$, knowing the input, checks the claim by evaluating $\tilde{V}_d(r_d)$ by itself. If all the checks passes, $\mathcal{V}$ accepts the result. Otherwise, reject.

We remark that the proof of soundness of GKR protocol is similar to that of sum-check protocol: if $\mathcal{P}$ have sent a wrong output to $\mathcal{V}$, then $\mathcal{V}$ gets the claim $\tilde{V}_i(r_i) = v_i'$ where $v_i'$ is different from the correct one $v_i$ for all $0 \le i \le d$ with overwhelming probability. As we briefly mentioned at the proof of Theorem 11, above argument follows from the Schwartz-Zippel lemma and the soundness of sum-check protocol.

### B.3 The GKR Protocol over Rings

[CCKP19] showed that, using the generalized Schwartz-Zippel lemma over rings, one can generalize the multilinear extension and Sum-Check protocol over rings (instead of fields), from which the generalized GKR protocol over rings follows. Here, ring is always a commutative ring with multiplicative identity 1.

**Schwartz-Zippel Lemma for Rings**. The first observation is that the Schwartz-Zippel Lemma also holds over the rings.

**Lemma 10 (Schwartz-Zippel for Rings).** *Let $R$ be a finite ring, and let $A \subset R$ be a finite set such that $\forall x, y \in A,\ x - y$ is invertible. We will call such $A$ a sampling set of $R$. Let $f : R^n \to R$ be an $n$-variate nonzero polynomial of total degree $D$. If $D \leq |A|$, then*

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

*Proof.* The proof can be found in many texts or in [CCKP19]. We first consider the case when $f$ is a univariate polynomial. If it has a root $a \in A$, then $f = (x - a) \cdot f_1$. If $b \in A$ is another root, then $f = (X - a)(X - b)f_2$ because $b - a$ is invertible and $b$ must be a root of $f_1$. By iterating $D$ times we prove that $f$ has at most $D$ roots in $A$. Extension to the multivariate case is the same as that of the original lemma (Lemma 8) using the induction. □

**Sum-check Protocol over Rings**. From the Schwartz-Zippel lemma over rings, one can naturally generalize the sum-check protocol over the rings.

**Theorem 12 (Sum-check Protocol over Rings [CCKP19]).** *Let $R$ be a finite ring with a sampling set $A \subset R$ (Lemma 10). For an $n$-variate polynomial $f : R^n \to R$ of degree at most $d \leq |A|$ in each variable and $\beta \in R$, the sum-check protocol is an interactive proof protocol to prove that $S(f) = \beta$, where*

$$S(f) := \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} f(b_1, b_2, \ldots, b_n).$$

*The protocol has the soundness probability $\frac{nd}{|A|}$. The computational complexity and the communication cost is the same as the original sum-check protocol (Theorem 12) except that the cost is measured by the number of operations (or elements) over $R$ instead of over $\mathbb{F}$.*

The protocol is the same as the original protocol over $\mathbb{F}$, except that the random points are chosen from the set $A$, which gives the soundness probability $\frac{nd}{|A|}$ instead of $\frac{nd}{|\mathbb{F}|}$. See [CCKP19] for the full proof and the description.

We remark that the polynomial interpolation (which is implicitly used in the sum-check protocol to identify and send a polynomial $f_i(t)$) is also possible in the ring $R$ with the elements of $A$ as a point of interpolation, since there difference is invertible (e.g., we can use the Lagrange interpolation method).

**Multilinear Extensions over Rings**. The multilinear extension can also be defined over rings, i.e., given a function $V : \{0,1\}^n \to R$, there exists a unique multilinear polynomial $\tilde{V}(\vec{x}) : R^n \to R$ extending $V$, i.e., $\tilde{V}(\vec{x}) = V(\vec{x})$ for all $\vec{x} \in \{0,1\}^n$. The construction of $\tilde{V}(\vec{x})$ is also the same as that over the field: see the equation following Lemma 9.

**GKR protocol over rings**. Since the Schwarz-Zippel lemma, sum-check protocol, and multilinear extension can be easily defined over rings, the GKR protocol can also be generalized over rings. In other words, for an arithmetic circuit $C : R^n \to R$ over a finite ring $R$ with fan-in 2, of size $S$ and depth $d$, given that the ring $R$ has enough size of sampling set $A$, the GKR protocol over ring is an interactive proof system that allows a prover $\mathcal{P}$ to prove that $C(x) = y$ to a verifier $\mathcal{V}$.

As in the case of sum-check, the only difference of the GKR protocol over rings from that over fields is that $\mathcal{V}$ samples random points from $A$ (instead of $R$), from which the soundness probability is modified to $\frac{7d\log S}{|A|}$ instead of $\frac{7d\log S}{|\mathbb{F}|}$ of the field case. We refer to [CCKP19] for the full description of the protocol and a detailed proof.

## C Hardness of LWE - Security of BV scheme

The BV scheme is secure if the corresponding LWE problem is hard (note that a LWE instance can be derived from a RLWE instance, so our analysis also applies to RLWE-based schemes). The hardness can be estimated based on known attacks. We will use $\sigma \simeq 3.2$ and $\alpha = \frac{\sqrt{2\pi}\sigma}{q} = \frac{8}{q}$, because it is used in practice. Here we give our detailed analysis of the harness of the LWE problem, based on state of the art estimates and attacks.

Let us first define the LWE problem: A LWE instance is a pair $(A, A^T s + e)$ where $A \in \mathbb{Z}_q^{n\times m}$ and $s \in \mathbb{Z}_q^n, e \in \mathbb{Z}_q^m$ (from the noise distribution, with standard deviation $\sigma$). The *Search*-LWE problem consists in finding the secret $s$ given a LWE instance. The *Decision*-LWE problem consists in distinguishing LWE instances from uniform pairs.

If $\sigma > 2\sqrt{n}$ then LWE is reduces to SVP on ideal lattices ([LPR10]), but in practice $\sigma$ is often smaller, for example, in SHE libraries such as SEAL or HElib $\sigma \simeq 3.2$ (which is the value used to attack them in [Alb17]).

The main security analysis comes from attacks which consist in finding one or several short vectors in some lattice by using lattice reduction: [LP11] analyses a decoding attack that solves the *Bounded-Distance Decoding* problem on the lattice $\Lambda(A^T) = \{z \in \mathbb{Z}^m : \exists s \in \mathbb{Z}_q^n,\ z = A^T s\ mod\ q\}$ to recover the secret from a short basis, [Alb17] uses a dual attack to solve the *Decision* problem by finding a short vector in the dual lattice $v \in \Lambda(A^T)^\perp = \{w \in \mathbb{Z}_q^m,\ w^T A^T = 0\ mod\ q\}$ and distinguishing $< v, e >$ from uniform. In both cases, the lattice reduction algorithm (BKZ) runtime nearly only depends on the root-Hermite factor defined as follows:

Let $v$ be the shortest vector outputed by BKZ on a lattice $L$, the the root-Hermite factor $\delta_0$ is defined by $||v|| = \delta_0^m |det(L)|^{\frac{1}{m}}$.

[LP11] finds an estimate runtime of

$$\log(t_{BKZ}(\delta)) = \frac{1.8}{\log\delta} - 110,$$

[APS15] updated it to

$$\log(\ t_{BKZ}(\delta_0)) = \frac{0.009}{\log^2\delta_0} + 4.1$$

Our security analysis for LWE will be based on the distinguishing attack from [Alb17], because it is one of the most recent ones, it is efficient against practical SHE libraries, and because it allows us to derive a security condition (as it did for the choice of parameters in [FV12], but it used the runtime estimate from [LP11], we will use the one from [APS15]).

A short vector of norm $||v||$ gives the attack an advantage $\varepsilon \simeq exp(-\pi\alpha^2||v||^2)$, with $\alpha = \frac{8}{q}$ ([LP11, Alb17]). So we need $||v|| = \frac{1}{\alpha}\sqrt{\frac{-ln(\varepsilon)}{\pi}} = \frac{F(\varepsilon)}{\alpha}$. [Alb17] shows how to run the attack several (e.g. $\frac{1}{\varepsilon^2}$) times with amortized costs to get a high advantage. In $\Lambda(A^T)^\perp$, the determinant is $q^n$ with high probability, so we get $||v|| = \delta_0^m q^{\frac{n}{m}}$. We will use the value $m = \sqrt{\frac{n\log q}{\log\delta_0}}$, which minimizes the norm of the short vector and gives the best attack. Thus, the distinguishing attack has advantage at least $\varepsilon$ if

$$\log\ \delta_0 \geq \frac{\log^2\left(\frac{1}{\alpha}F(\varepsilon)\right)}{4n\log q}$$

where $F(\varepsilon) = \sqrt{\frac{-ln(\varepsilon)}{\pi}}$.

For the attack runtime to be $2^\lambda$, we need $\log\ \delta_0 = \sqrt{\frac{0.009}{\lambda - 4.1}}$, and finally we get a security condition for LWE:

$$\sqrt{\frac{0.009}{\lambda - 4.1}} = \log\ \delta_0 \geq \frac{\log^2\left(\frac{1}{\alpha}F(\varepsilon)\right)}{4n\log q} \tag{8}$$

If we chose a security parameter $\lambda = 128$, an advantage $\varepsilon = 2^{-64}$ for the distinguishing attack, and $\alpha = \frac{8}{q}$ as usual HE scheme, (8) gives $\log\delta_0 \simeq 0.008523$ and $\frac{1}{\alpha}F(\varepsilon) = q\frac{F(\varepsilon)}{8} \simeq 0.4697q$. Then, using a small approximation to simplify the expression:

$$\frac{\log^2\left(\frac{F(\varepsilon)}{\alpha}\right)}{4d_f\log q} = \frac{1}{4d_f}\left(\log q + \log(0.4697) + \frac{\log^2(0.4697)}{\log q}\right) \simeq \frac{1}{4d_f}(\log q - 1)$$

Finally (8) becomes:

$$\log q \leq 4d_f\log\delta_0 + 1 \simeq \frac{d_f}{30} + 1 \tag{9}$$

The security condition was obtained with some approximations, but we used a conservative estimation of the attack runtime, and we can eventually check that the required security is well achieved using the estimator [APS15] (`https://bitbucket.org/malb/lwe-estimator`) which estimates the runtime of various attacks on given LWE problem with the parameters $n, q, \alpha$.

## D Generalization to an Arbitrary Modulus $q$

Our VC scheme (Section 3, Section 4) also works for an SHE scheme with $q$ not a power of a prime, i.e., when $q$ is a product of distinct primes. Let $q = \prod_{i=1}^N p_i^{e_i}$, then the Chinese Remainder Theorem (CRT) gives that

$$\mathbb{Z}_q[X] \ \cong \mathbb{Z}_{p_1^{e_1}}[X] \times ... \times \mathbb{Z}_{p_N^{e_N}}[X],$$

and running our VC over the ring $\mathbb{Z}_q[X]$ (or $\mathbb{Z}_q[X]/(f)$) is equivalent to running it over each subring $\mathbb{Z}_{p_i^{e_i}}[X]$(or $\mathbb{Z}_{p_i^{e_i}}[X]/(f_i)$), $i \in \{1, 2, \ldots, N\}$ whose modulus is a prime power, which is the case given in our proposal. The verifier $\mathcal{V}$ accepts the result only if it accepts all the proofs (total $N$) on each ring, and a (dishonest) $\mathcal{P}$ can cheat only if he can break the soundness of the argument system over at least one of the subrings, whose probability is bounded by $O(\sum_{i=1}^N 1/p_i^{d_{h_i}})$ where $d_{h_i}$ is the degree of each Galois ring $\mathbb{Z}_{p_i^{e_i}}[X]/(h_i)$ for the homomorphic hash function (Section 4.3) on each subring $\mathbb{Z}_{p_i^{e_i}}[X]$.

We additionally remark that our VC is appealing in its efficiency (especially, in $\mathcal{P}$'s cost) also when working with a circuit over $\mathbb{Z}_q[X]/(f) \cong \prod_{i=1}^{N} \mathbb{Z}_{q_i}[X]/(f)$ where $q_i$ is a small prime (less than $2^\lambda$) and $f$ fully splits over $\mathbb{Z}_{q_i}$,[24] i.e., $\mathbb{Z}_{q_i}[X]/(f) \cong (\mathbb{Z}_{q_i})^M$ where $M$ is the degree of $f$. As mentioned at the above, to run VC over $\mathbb{Z}_q[X]/(f)$, one should run the VC on each subring $\mathbb{Z}_{q_i}[X]/(f)$, and it can be boosted by our method. For concrete analysis, let $C$ be an arithmetic circuit over $\mathbb{Z}_{q_i}[X]/(f)$, and let $S$ and $D$ be the size and the degree of $C$, respectively. We measure the cost by counting the number of operations over $\mathbb{Z}_{q_i}$ required. Then, the cost of $\mathcal{P}$ for computing and proving the evaluation of $C$ is as follows:

In a naive method (without our homomorphic hash functions), one can exploit the CRT isomorphism $\mathbb{Z}_{q_i}[X]/(f) \cong (\mathbb{Z}_{q_i})^M$ then apply the VC over each subring $\mathbb{Z}_{q_i}$ (total $M$ times). Then, to achieve $1/2^\lambda$ soundness, the cost of $\mathcal{P}$ for getting the result and for generating the proof are as follows (the former is denoted by $\mathcal{P}_{comp}$ and the latter is by $\mathcal{P}_{proof}$):

- $\mathcal{P}_{comp} = O(MS)$
- $\mathcal{P}_{proof} = O(\frac{\lambda}{\log q_i} \cdot MS)$ ($\because$ needs to repeat the proof generation $\frac{\lambda}{\log q_i}$-times.)

In contrast, in our method with homomorphic hash functions, $\mathcal{P}$ evaluates the circuit $C$ without modulo $f$, then proves the computation over a ring $\mathbb{Z}_{q_i}[X]/(h)$ where the degree of $h$ is set to be $\frac{\lambda}{\log q_i}$ to achieve $1/2^\lambda$ soundness. Then, the cost of $\mathcal{P}$ follows ($\tilde{O}$ hides logarithmic factors):

- $\mathcal{P}_{comp} = \tilde{O}(DMS)$
- $\mathcal{P}_{proof} = \tilde{O}(\frac{\lambda}{\log q_i} \cdot S)$

Therefore, our method is more efficient when the degree $D$ of circuit is not very large while the degree $M$ of $f$ is very large, which is exactly the case of SHE schemes.

# E Proofs

## E.1 Ben-Or's Generation of Irreducible Polynomials

We describe Ben-Or's algorithm to generated an irreducible polynomial. We slightly change the original algorithm in order to make explicit the amount of random coins that it needs in order to make its probability of failing negligible. Our change also makes clear how it can be implemented on input public random coins.

**Theorem 13 (Ben-Or's Generation of Irreducible Polynomials [Ben81]).** *The following algorithm uniformly samples a monic irreducible polynomial of degree $d_h$ in $\mathbb{Z}_p[X]$ taking expected number of $\tilde{O}(d_h^2 \log p)$ operations in $\mathbb{Z}_p$ ($\tilde{O}$ hides logarithmic factors in $d_h$). Furthermore, for $N = 2d_h(d_h - 1)\lambda$, the algorithm outputs* `fail` *with probability $\leq 2^{-\lambda}$.*

[Algorithm] ('gcd' denotes the greatest common divisor)

- Input: A prime $p$, a degree $d_h \in \mathbb{Z}_{>0}$ and a security parameter $\lambda \in \mathbb{Z}_{>0}$.
- Random coins: $\rho_1, \ldots, \rho_N \in \mathbb{Z}_p$.
- Output: A monic irreducible polynomial of degree $d_h$ in $\mathbb{Z}_p[X]$.
- Procedure:
  1. Initialize $j = 0$.
  2. If $j = 2d_h\lambda$ return `fail`

[24] It arises commonly in many HE schemes [BEHZ16, HPS19, CHK+18] aiming faster implementation exploiting CRT (Chinese Remainder Theorem) and NTT (Number Theoretic Transform).

3. Set $k = j \cdot (d_h - 1)$, and build a monic polynomial $h \in \mathbb{Z}_p[X]$ of degree $d_h$ by using $(\rho_{k+1}, \ldots, \rho_{k+d_h-1})$ to define its coefficients.
4. for $i = 1, \ldots, \lfloor \frac{d_h}{2} \rfloor$ do
   if $\gcd(h, X^{p^i} - X) \neq 1$, set $j \leftarrow j + 1$ and goto 2.
5. return $h$

*Proof.* See [VZGG13, Theorem 14.42] for the full proof. The correctness follows since every monic irreducible polynomial of degree $d$ divides $X^{p^d} - X$. Hence, a monic polynomial $h$ of degree $d_h$ is reducible if and only if $\gcd(h, X^{p^i} - X) \neq 1$ for some $1 \leq i \leq \lfloor \frac{d_h}{2} \rfloor$. The complexity follows from the fact that randomly sampled $h$ is irreducible with probability greater than $\frac{1}{2d_h}$ (there exists more than $\frac{p^{d_h}}{2d_h}$ (Lemma 4) monic irreducible polynomials among $p^{d_h}$ candidates). To bound the probability of failing, this is the probability that at each step $j$ the random coins do not yield an irreducible polynomial. Let us call $\mathsf{bad}$ the failing event, then (given the independence of the coins used in each step) we have $\Pr[\mathsf{bad}] = \prod_{j=1}^{2d_h\lambda} \Pr[\mathsf{bad}_j]$, where $\mathsf{bad}_j$ is the event that the polynomial built in step $j$ is not irreducible, and the probabilities are taken over the random choice of $\vec{\rho} \xleftarrow{\$} \mathbb{Z}_p^N$. As mentioned above, for every step $j$, $\Pr[\mathsf{bad}_j] \leq (1 - \frac{1}{2d_h})$, and thus $\Pr[\mathsf{bad}] \leq (1 - \frac{1}{2d_h})^{2d_h\lambda} \leq \frac{1}{e^\lambda}$. □

*Remark 16 (Lower Bound of $N$).* In the above proof, the probability of random sampled $h$ being irreducible is upper bounded by $\frac{1}{d_h}$ ([VZGG13, Lemma 14.38]). With this, $\Pr[\mathsf{bad}] = \prod_{j=1}^{(d_h-1)\lambda'} \Pr[\mathsf{bad}_j] \geq (1 - \frac{1}{d_h})^{(d_h-1)\lambda'} \geq \frac{1}{e^{\lambda'}} \geq \frac{1}{2^\lambda}$ when $\lambda' \leq \lambda \ln 2$. Therefore, we can see that, in the above theorem, $N$ must be greater than $(\ln 2)(d_h - 1)^2\lambda$ to achieve negligible failure probability.

### E.2 Proof of Theorem 6

We want to argue three points: first, that (conditioned to not aborting) the algorithm yields a uniform monic irreducible polynomial of degree $d_h$; second, that the algorithm aborts with negligible probability; and third, that the complexity is as promised.

**Lemma 11.** *Let $\alpha$ be a generator of $\mathbb{F}^\times_{p^{d_h}}$, and let $k | d_h$, $k \neq d_h$. Then for $0 \leq j < p^{d_h} - 1$, $\alpha^j$ is in the proper subfield $\mathbb{F}_{p^k}$ if and only if $j = \sum_{\ell=0}^{d_h-1} a_\ell p^\ell$ where $(a_0, a_1, \ldots, a_{d_h-1}) \in \mathsf{Bad}_k$.*

*Proof.* First of all, $\alpha^j$ is in $\mathbb{F}_{p^k}$ if and only if $(\alpha^j)^{p^k-1} = 1$. From here and the fact that $\alpha$ is a generator of $\mathbb{F}^\times_{p^{d_h}}$, we get that $\alpha^j$ is in $\mathbb{F}_{p^k}$ if and only if $j = i \cdot \frac{p^{d_h}-1}{p^k-1}$, where $i \in \{0, \ldots, p^k - 2\}$. However $\frac{p^{d_h}-1}{p^k-1} = 1 + p^k + p^{2k} + \cdots + p^{(\frac{d_h}{k}-1)k}$.

This means that if the $p$-ary decomposition of $i$ is $i = i_0 + i_1 p + \cdots + i_{k-1}p^{k-1}$ (with $0 \leq i_\ell \leq p-1$), then the $p$-ary decomposition of $j = i \cdot \frac{p^{d_h}-1}{p^k-1}$ i.e., $j = \sum_{\ell=0}^{d_h-1} a_\ell p^\ell$ is given by $a_\ell = i_{(\ell \mod k)}$ for all $\ell = 0, \ldots, d_h - 1$, which is equivalent to being in $\mathsf{Bad}_k$, where $\mathbf{v} = (i_0, i_1, \ldots, i_{k-1})$. □

**Lemma 12.** *Conditioned to the algorithm in Theorem 6 not aborting, its output is a polynomial distributed uniformly randomly in $I(d_h, p)$, the set of monic irreducible polynomials in $\mathbb{Z}_p[X]$ of degree $d_h$*

*Proof.* By the theory of finite fields, the minimal polynomial in $\mathbb{Z}_p[X]$ of $\beta \in \mathbb{F}^{\times}_{p^{d_h}}$ (which is irreducible, monic and unique) has degree equal to the smallest $k \geq 1$ for which $\beta$ belongs to the subfield $\mathbb{F}_{p^k}$ (and this $k$ must divide $d_h$). Denote by $S$ the set of elements $\beta \in \mathbb{F}^{\times}_{p^{d_h}}$ which are not in any proper subfield. If $\beta \in S$, it must be the case that its minimal polynomial is of degree $d_h$, i.e. it is in $I(d_h, p)$. On the other hand, every polynomial in $I(d_h, p)$ has exactly $d_h$ different roots in $S$. Therefore mapping an element to its minimal polynomial induces a $d_h - to - 1$ map between $S$ and $I(d_h, p)$.

Note that by the preceeding lemma, if the algorithm does not abort, it clearly samples $\beta$ uniformly at random in $S$ either in step 1 or 2, and therefore the output of the algorithm is uniform in $I(d_h, p)$.

**Lemma 13.** *The algorithm in Theorem 6 fails with probability at most* $\frac{4}{p^{d_h-1}}$. [25]

*Proof.* With notation as in the previous lemma, note that the set $S$ has size $d_h \cdot I(d_h, p)$, which by Lemma 4 is of size at least $p^{d_h} - 2p^{\lfloor \frac{d_h}{2} \rfloor}$. Every element in $S$ is realized as $\beta$ in step 1 for exactly one possible coin-vector $(\rho_0^{(1)}, \rho_1^{(1)}, \ldots, \rho_{d_h-1}^{(1)})$ (the same holds for step 2 and $(\rho_0^{(2)}, \rho_1^{(2)}, \ldots, \rho_{d_h-1}^{(2)})$)

The probability that step 1 outputs an element in $S$ and continues to step 4 is hence $|S|/p^{d_h} \geq 1 - 2p^{-\lfloor \frac{d_h}{2} \rfloor}$, and the same holds for step 2 (if the algorithm enters that step). Therefore the probability that the algorithm fails is at most $(2p^{-\lfloor \frac{d_h}{2} \rfloor})^2 \leq \frac{4}{p^{d_h-1}}$.

*Remark 17.* If $d_h$ is prime then $|S| = p^{d_h} - p$ and step 1. already outputs some $\beta$ with overwhelming probability $1 - p^{d_h-1}$. Therefore in that case we can omit step 2. and the algorithm still fails with negligible probability.

**Lemma 14.** *The algorithm in Theorem 6 takes* $O(d_h M(d_h) \log p)$ *operations in* $\mathbb{Z}_p$.

*Proof.* Consider first steps 1. and 2. The algorithm needs to check whether $(\rho_0^{(b)}, \ldots, \rho_{d_h-1}^{(b)}) \notin \bigcup_{k|d_h, k \neq d_h} \mathtt{Bad}_k$ (where $b$ denotes the step number) and compute $\beta$ if this is not the case. For the former, note that if $k_1 | k_2$, then $\mathtt{Bad}_{k_1} \subseteq \mathtt{Bad}_{k_2}$. Hence it is enough to check that $(\rho_0^{(b)}, \ldots, \rho_{d_h-1}^{(b)}) \notin \bigcup_{p \text{ prime}, p|d_h} \mathtt{Bad}_{d_h/p}$, because every proper divisor of $d_h$ must be a divisor of $d_h/p$ for some prime divisor $p$ of $d_h$. Each of the tests whether $(a_0, \ldots, a_{d_h-1}) \in \mathtt{Bad}_{d_h/p}$ takes $d_h$ comparisons in $\mathbb{Z}_p$, as it just checks $a_i = a_{i \mod d_h/p}$ for all $i$. On the other hand, $d_h$ has at most $\log d_h$ distinct prime divisors and hence the test in step 1 (resp. step 2) takes $O(d_h \log d_h)$ comparisons in $\mathbb{Z}_p$. Next computing $\beta$ requires computing $d_h$ exponentiations in $\mathbb{F}_{p^{d_h}}$ where the exponent is at most $p-1$, and then $d_h - 1$ multiplications in $\mathbb{F}_{p^{d_h}}$. Each of the aforementioned exponentiations takes $O(\log p)$ multiplications in $\mathbb{F}_{p^{d^h}}$, and each such multiplication consists of $M(d_h)$ operations in $\mathbb{Z}_p$, so the total complexity of the computation of $\beta$ is $O(d_h M(d_h) \log p)$ operations. We remark that this step gives a $\beta$ expressed as $p(\alpha)$, where $p \in \mathbb{Z}_p[X]$ of degree $< d_h$. Finally we consider step 4. In this case we use an algorithm by Shoup[Sho99] that takes $\beta$ expressed in the form explained above, and computes its minimal polynomial in $O(d_h^{1/2} M(d_h) + d_h^2)$ operations over $\mathbb{Z}_p$. Since clearly $d_h \leq M(d_h)$, the cost is dominated by the computation of $\beta$ in step 1 (or 2), which is $O(d_h M(d_h) \log p)$ operations in $\mathbb{Z}_p$. □

[25] This is in fact not too tight, as it comes from the bound in Lemma 4 which is quite rough, but it is good enough for our purposes.

### E.3 Proof of Theorem 7

*Proof.* The theorem is proved by the following complexity analysis on each building block.

**Description of Circuit $C'$ with BV scheme.**
We first bound the size $S'$ and the depth $d'$ of the circuit $C'$ (over $R_q$) representing the prover $\mathcal{P}$'s computation $\hat{g} : (R_q^D)^n \to \mathbb{Z}_q[X]^D$ on the ciphertexts. Recall that each ciphertext is composed of at most $D$ elements of $R_q$. Hence, each plaintext addition corresponds to $O(D)$ operations over $R_q$ while each plaintext multiplication corresponds to $O(D^2)$ operations over $R_q$. To estimate the cost of the GKR protocol (esp. for the verifier), the depth of the circuit should be considered, and we can assume that both addition and multiplication corresponds to a circuit of depth $O(1)$.[26] Therefore, the size and depth of the circuit $C'$ over BV ciphertexts are respectively, $S' = O(D^2 S)$ and $d' = O(d)$. Note that the prover sends the result ciphertext (without modulo reduction by $f$) to the verifier, which appears as the $O(D^2 d_f)$ term in the communication cost (in the number of elements of $\mathbb{Z}_q$).

**Cost of Hashing.**
We consider two cases using (i) single hash $H_h$ and (ii) double hash $H_{r,h}$. Sampling $h \in \mathbb{Z}_q[X]$ can be done efficiently (with $\tilde{O}(d_h^2 \log p)$ or $O(d_h M(d_h) \log p)$[27] operations in $\mathbb{Z}_p$, see Section 4.3.3) by $\mathcal{V}$ (and $\mathcal{P}$ in a non-interactive argument). Note that it suffices to take $d_h \approx \lambda / \log p$ for $2^{-\lambda}$ soundness, and the cost for sampling is negligible compared to the complexity required in the following task. Now, the prover and verifier evaluate the input and output ciphertext under the hash function $H_h$ or $H_{r,h}$ with $\tilde{O}((n + D^2) d_f)$ operations in $\mathbb{Z}_q$,[28] which gives such cost in $T_{\mathcal{P}}$ and $T_{\mathcal{V}}$. We finally note that $H_h$ maps each addition and multiplication of $C'$ to each addition and multiplication over $\mathbb{Z}_q[X]/(h)$, while $H_{r,h}$ maps $O(D)$ operations (or $O(D^2)$ operations) of $C'$ corresponding to one addition (resp. one multiplication) of $C$ to one addition (resp. one multiplication) over $\mathbb{Z}_q[X]/(h)$. It is significant when considering the cost of GKR protocol in the following.

**Complexity of GKR.**
Recall that the complexity of GKR protocol on the circuit (over $\mathbb{Z}_q[X]/(h)$) of size $S$ and depth $d$ with $n$ input is as follows (time complexity of $\mathcal{P}$ or $\mathcal{V}$; communication cost):

$$(T_{\mathcal{P}}^{\Pi}, T_{\mathcal{V}}^{\Pi}, C_{\mathcal{C}}^{\Pi}) = (O(S),\ O(n + d \log S),\ O(d \log S))$$

where the measure of complexity is the number of operations or elements in $\mathbb{Z}_q[X]/(h)$. Therefore, plugging this equation into the case of single hash where the circuit (to be proved and verified) is over $\mathbb{Z}_q[X]/(h)$ of size $S' = O(D^2 S)$ and depth $d' = O(d)$, then multiplying $\tilde{O}(d_h) = \tilde{O}(\lambda)$[29] to each cost since we count the complexity in number of operations (or elements) in $\mathbb{Z}_q$ (instead of $\mathbb{Z}_q[X]/(h)$), we get the desired result. Similarly, in case of double hash, since the circuit (to be proved and verified) is over $\mathbb{Z}_q[X]/(h)$ of size $O(S)$ and depth $O(d)$, we get the result. □

[26] Computing each coefficient accompanies evaluating a tree of binary addition gates with $O(D)$ inputs, and verification on it can be done as if it is a depth 1 layer having gates summing multiple inputs (see [Tha13, (Full version) Section 5]).

[27] $M(d_h)$ denotes the complexity of multiplying polynomials of degree $d_h$.

[28] Dividing a polynomial of degree $a$ by another one of degree $b$ takes $\tilde{O}(a - b)$ operations ([VZGG13])

[29] We assume that each multiplication of $\mathbb{Z}_q[X]/(h)$ can be done in $\tilde{O}(d_h)$ operations of $\mathbb{Z}_q$ with fast multiplication methods (e.g., FFT).