Implementing Predicate Encryption for Branching Programs: Private Computation over Key-Homomorphic Encodings Without FHE

Sora Suegami[0009-0008-7107-3676]

Ethereum Foundation sora.suegami@ethereum.org

Abstract

Predicate encryption (PE) encrypts a message under an attribute that the ciphertext hides, allowing a secret key for a predicate to recover the message only when the attribute satisfies the predicate. Existing public-key implementations support useful specialized predicates, but enforcing heterogeneous conditions with one key requires an attribute representation fixed before encryption, which grows with the supported types of conditions and cannot express types it does not anticipate, such as regular expressions. Lattice-based PE for general circuits is far more expressive, but existing constructions rely on fully homomorphic encryption (FHE), typically evaluated as a Boolean circuit over key-homomorphic encodings, which expands the computation and amplifies noise growth and leads to impractically large parameters. Existing PE thus offers either practicality for restricted predicates or expressiveness at an impractical cost.

We present a lattice-based PE scheme for branching programs that avoids FHE entirely and combines heterogeneous conditions in a single predicate over a common attribute representation. It rests on two techniques: asymmetric multiplication, which multiplies a hidden program state by an encrypted attribute symbol through decryption inside an encoding, and lookup tables over exponents, which evaluate arbitrary state transitions on a single encoding by ring automorphisms. Under circular Ring-LWE, the scheme is selectively weakly attribute-hiding against polynomially many colluding unauthorized keys.

We give, to our knowledge, the first implementation of PE for a predicate class as expressive as branching programs. On a single NVIDIA A100 GPU, for a predicate that combines a range check and a regular-expression check on a 10-bit attribute, with parameters targeting 128-bit security against known Ring-LWE attacks, encryption takes 4.06 hours, key generation 4.70 hours, and decryption 4.23 hours, with 342.25 GB ciphertexts. These costs remain large, but compared with a single FHE multiplication evaluated as a Boolean circuit over encodings, the basic step of PE that evaluates FHE over encodings, whose ciphertexts alone occupy tens of exabytes per attribute bit, our ciphertexts are at least 2×10^9 times smaller per attribute bit, and our decryption needs at least 2×10^{16} times fewer 64-bit integer multiplications.

This work was carried out with the assistance of GPT-6 Astra and Claude Opus 5.5, as detailed at the end of the introduction.

1 Introduction

Predicate encryption (PE) [KSW08] associates a ciphertext with an attribute x and a message m, and a secret key with a predicate. A key is authorized for an attribute if its predicate accepts that attribute, and unauthorized otherwise. An authorized key recovers m, and, unlike in attribute-based encryption (ABE), the ciphertext hides x. PE thus enables fine-grained access control over encrypted data without revealing the attributes on which access decisions are based.

Practical PE for restricted predicates. Public-key PE has been implemented for inner-product predicates [Par11, Sco11] and conjunctions of equality tests with wildcards [IP08, AGM+13], enabling polynomial zero tests [KSW08] and conjunctive comparison, range, and subset queries [BW07]. Each of them, however, requires the encryptor to fix an attribute representation tailored to the type of predicate, such as indicator vectors over the domain for subset queries.

Practical predicates often combine heterogeneous conditions. For instance, an auditor of confidential audit records may be authorized to decrypt a record only if its numerical attributes fall within given ranges *and* its event sequence matches a regular expression describing a newly discovered attack. Separate keys for the two conditions do not enforce their conjunction, since either key alone reveals the record. A single key, in turn, requires a representation that supports both types of conditions: in existing schemes, the representation can grow as more types are supported, even when the underlying input is unchanged [BW07], and the schemes cannot support types, such as regular expressions, that the representation does not anticipate.

Expressive PE for general circuits. In contrast, PE for general circuits has been known from lattices since the work of Gorbunov, Vaikuntanathan, and Wee [GVW15], but these constructions are far from practical. A prominent line of constructions [GVW15, BTVW17, WWLW25] relies on fully homomorphic encryption (FHE) and the key-homomorphic BGG+ encodings [BGG+14]: the encryptor encrypts the attribute under the FHE scheme of Gentry, Sahai, and Waters (GSW) [GSW13] and encodes the bits of the FHE ciphertext, and decryption evaluates the FHE evaluation algorithm, represented as a Boolean circuit, over these encodings.

This layered approach introduces substantial computational overhead. Implementing FHE operations as Boolean circuits over key-homomorphic encodings substantially increases the number of encoding operations (*circuit expansion*). Moreover, the increased evaluation depth amplifies noise growth, potentially requiring a much larger modulus and lattice dimension, and larger parameters in turn make each encoding operation more expensive (*parameter amplification*). Consequently, the overall computational cost reflects both the increased number of operations and their increased individual cost. The required parameters can also make the ciphertext and key sizes prohibitively large, since they enlarge every encoding and matrix.

Generic transformations from ABE to PE are also known via lockable obfuscation [GKW17, WZ17]. In their general-purpose instantiations, these constructions use FHE to homomorphically evaluate the ABE decryption algorithm. The resulting FHE ciphertext is then processed by a low-depth lockable obfuscator based on lattice encodings in the style of the graph-induced multilinear maps of Gentry, Gorbunov, and Halevi [GGH15]. Thus, although this approach avoids simulating FHE over BGG+ encodings, it still incurs the substantial overhead of homomorphically evaluating ABE decryption under FHE. Hsieh, Lin, and Luo [HLL23] obtain PE for circuits of unbounded depth in this way, from a depth-unbounded ABE obtained by evaluating GSW computation over BGG+ encodings; their PE therefore depends on both forms of FHE evaluation.

Existing PE schemes thus trade expressiveness against practicality: those that have been implemented support only restricted predicates, whereas those for general circuits, which are known from lattices, rely on FHE, and their computational costs and data sizes remain a major obstacle to computational feasibility. This state of affairs leads to the central question of this work:

Can PE for expressive, heterogeneous predicates be made concretely feasible?

1.1 Our Contributions

Our answer is affirmative: we construct a new public-key PE scheme for arbitrary polynomial-length branching programs, without FHE or lockable obfuscation. Branching programs are expressive

Table 1: Our scheme on a whole predicate, compared with a baseline that evaluates only a single AND of two encrypted attribute bits, at 128-bit security. In PE schemes that evaluate FHE over BGG+ encodings [GVW15, BTVW17, WWLW25], this AND is one Ring-GSW multiplication over BGG+ encodings, a common bottleneck of these schemes. The nonlinear predicate considered here requires at least one multiplication in this straightforward FHE implementation, so the baseline is a deliberately minimal lower bound for this implementation. Our scheme evaluates a predicate that combines a range check and a regular-expression check on a 10-bit attribute (Section 7.1). The baseline is implemented but cannot be run end to end at these parameters, so its costs are estimated and its running times are omitted. The running times of our scheme are measured end to end on a single NVIDIA A100 GPU. Sizes of mpk and ct are per attribute bit, with all public-key matrices expanded for both schemes; "Mul" and "Decomp." count the 64-bit integer multiplications and digit decompositions in decryption.

			Size per attribute bit		Decryption ops		Time		
Scheme	Modulus size (bits)	Ring dimension	mpk	ct	Mul	Decomp.	KeyGen	Enc	Dec
Baseline: a single 1-bit AND (one GSW mult. over BGG+)	8,184	2^{19}	139.24 EB	69.62 EB	2.32×10^{30}	1.54×10^{25}	–	–	–
Ours: range and regex checks (10-bit attribute)	694	2^{15}	67.65 GB	34.23 GB	8.70×10^{13}	8.30×10^{10}	4.70 h	4.06 h	4.23 h

enough for many practical policies: they naturally capture computations that read the attribute symbol by symbol while maintaining a small state, including regular-expression pattern matching via finite automata, comparisons and ranges on integers, Hamming-distance thresholds, and, by Barrington's theorem [Bar89], all Boolean formulas. They can also combine such heterogeneous conditions within a single program: conditions on the same input symbol can be incorporated into one transition without increasing the program size. Our scheme therefore enforces heterogeneous predicates with a single secret key, using a common attribute representation independent of the predicates chosen after encryption, within the bounds on the length, width, and alphabet fixed at setup.

In the key-policy setting that we consider, the state transition rules of the branching program are public and associated with a secret key, whereas the attribute symbols read at each step are private and associated with a ciphertext. Assuming circular Ring-LWE with automorphism keys, the scheme is selectively secure and weakly attribute-hiding: the ciphertext hides its attribute and message even from adversaries that collude with polynomially many keys, as long as none of these keys is authorized for the attribute.

Implementation and evaluation. We implement the scheme on GPUs. On a single NVIDIA A100 GPU, we evaluate its concrete costs for a heterogeneous predicate that imposes both a regular expression and a range on the same 10-bit attribute, a calendar date that must be valid and lie in a date window (Table 1).

As a baseline, we consider a single multiplication of the ring variant of GSW, represented as a Boolean circuit. We estimate the cost of evaluating this circuit over BGG+ encodings, which dominates decryption in PE schemes that evaluate FHE over BGG+ encodings [GVW15, BTVW17, WWLW25] and lower-bounds their cost for the nonlinear predicate considered here in this straightforward FHE implementation, which requires at least one multiplication. For the whole predicate, our scheme uses a modulus at least 11 times shorter, ciphertexts at least 2×10^9 times smaller per attribute bit, and at least 2×10^{16} times fewer 64-bit integer multiplications in decryption than this single multiplication.

Results on the concrete efficiency for branching programs of different scales will be added in

a future version. These results bring expressive lattice-based PE from parameters that cannot be instantiated, with ciphertexts of exabytes per attribute bit, to sizes that current hardware can process. This improvement stems from evaluating the branching program directly over the encodings, instead of evaluating FHE as a Boolean circuit over them.

Technical approach. Our scheme rests on two new techniques: *asymmetric multiplication*, which multiplies a hidden state by a hidden attribute symbol without FHE, and *lookup tables over exponents*, which compute arbitrary state transitions on a single encoding. BGG+ encodings multiply two encoded values as long as one of them is public, while the other may remain hidden. For general circuits, both factors of a multiplication may be intermediate values, so that prior schemes make every multiplied value public as bits of an FHE ciphertext and evaluate FHE gate by gate over the encodings. Our key observation is that a branching program never needs this: the output of every multiplication, the new state, is used only as the hidden factor of the next multiplication, whose public factor is always derived from a fresh attribute symbol. We therefore supply each symbol as a gadget ciphertext, whose digits are public as in prior schemes, and keep the state hidden in its encoding. This asymmetric multiplication of the state encoding by these digits yields an encoding of the product, which serves as the hidden next state; over the encodings, only FHE decryption takes place, and no FHE multiplication.

Since the state is never multiplied by itself, every transition must then be computed by operations that never multiply the state by itself, such as additions and multiplications by public ring elements. Our second technique, lookup tables over exponents, expresses arbitrary transitions in this way while avoiding the growth in data size caused by a one-hot representation of the state: we represent states as monomial exponents, so that one encoding holds the whole state, and evaluate each transition as a lookup table using ring automorphisms applied homomorphically over the encodings. Section 2 explains both techniques.

Limitations and future work. Our scheme is only weakly attribute-hiding, so that it provides no security against an adversary that obtains a key authorized for the challenge attribute. Such authorized keys are in fact harmful: as in the constructions of [GVW15, BTVW17], decryption with an authorized key leaks the noise in the encodings of the private attribute and thereby the secret [Agr17].

In terms of efficiency, every step multiplies the noise by a large factor, so that the modulus grows exponentially in the length of the program, and long programs cannot be instantiated with practical lattice parameters. Moreover, our ciphertexts remain large even for small branching programs, at hundreds of gigabytes for a 10-bit attribute. Our scheme is therefore currently limited to settings with few ciphertexts or with large payloads associated with each attribute, such as encrypted archives. Applications that encrypt many attributes separately, such as log filtering, require ciphertexts several orders of magnitude smaller, which we leave to future work.

Paper organization. The rest of this section reviews related work in Subsection 1.2. Section 2 gives a technical overview of asymmetric multiplication and lookup tables over exponents. Section 3 recalls notation, lattice preliminaries, circular Ring-LWE, the definition of predicate encryption, and branching programs, together with examples such as regular expressions. Section 4 develops private computation over key-homomorphic encodings: the primitive operations over encodings, lookup tables, and the evaluation of branching programs. Section 5 presents the PE scheme with its parameters, proves its correctness, and states its security. Sections 6 and 7 describe our GPU

Table 2: Lattice-based PE. "Keys" gives the number of unauthorized / authorized keys for the challenge attributes that an adversary may obtain; Q is an a-priori bound. "FHE-free" indicates that the construction does not rely on FHE evaluation. Unless explicitly marked as depth-unbounded, "circuits" means circuits of a-priori bounded depth. For PE from lockable obfuscation [GKW17, WZ17], the underlying ABE is assumed to be the lattice-based construction of [BGG+14]. †Circular Ring-LWE with automorphism keys (Assumption 1).

Scheme	Predicates	Keys	Security	Assumption	FHE-free
[AFV11]	inner products	poly / 0	selective	LWE	yes
[GMW15]	range queries	poly / 0	selective	LWE	yes
[Kat17]	multi-dimensional equality	poly / 0	selective	LWE	yes
[GVW15]	circuits	poly / 0	selective	LWE	no
[BTVW17]	circuits	poly / 0	selective	LWE	no
[WWLW25]	circuits	poly / Q	semi-adaptive	LWE	no
[GKW17, WZ17]	circuits	poly / 0	selective	LWE	no
[HLL23]	circuits (depth-unbounded)	poly / 0	selective	evasive circular LWE	no
This work	branching programs	poly / 0	selective	circular RLWE†	yes

implementation and its experimental evaluation. Appendix A proves the security of the scheme, and Appendix B gives its variant with compressed digit encodings.

1.2 Related Work

Lattice-based PE. Table 2 summarizes representative lattice-based PE. Without FHE, PE is known for inner products [AFV11], range queries [GMW15], and multi-dimensional equality [Kat17]. For circuits, Gorbunov, Vaikuntanathan, and Wee [GVW15] encrypt the attribute under FHE and combine it with BGG+ encodings: decryption computes the FHE decryption over the encodings as an inner product with the FHE secret key, followed by the lazy-OR trick, which checks whether the result decrypts to 0 without rounding it. Brakerski et al. [BTVW17] avoid both the inner product with an encoded FHE secret key and the lazy-OR trick by the dual-use technique, in which the FHE secret key coincides with the secret of the encodings. Lai, Liu, and Wang [LLW21] handle a bounded number of authorized keys by two-stage sampling, and Wang et al. [WWLW25] combine it with the dual-use technique. Single-key and bounded-collusion functional encryption [GKP+13, AR17] hides the attribute fully, but only against an a-priori bounded number of keys, which includes unauthorized keys. Among the schemes that support branching programs without an a-priori bound on the number of unauthorized keys, ours is the only FHE-free one.

Efficiency comparison. Table 3 compares asymptotic sizes and costs for a branching program of length T and width w over an alphabet Σ.[1] Since the state occupies a single encoding, a step

[1]We count word operations, with $\mathbb{Z}_q$ in RNS form, so that costs reflect the size of q, and use $\ell \le T$, since every symbol is read at least once. For FHE-based schemes, a symbol is encrypted in binary with $\log|\Sigma|$ bits and expanded into a one-hot vector u inside the evaluated circuit, the state is a one-hot vector, the step of a symbol x is the $w \times w$ Boolean transition matrix $\sum_v u_v \mathbf{M}_v$, which needs only $w|\Sigma|$ additions, and the program is evaluated as a balanced tree of products of these matrices, of depth $O(\log T \cdot \log w)$, in which a step amortizes w^3 AND gates. For our scheme, using the convention of Table 3, $\log q = \widetilde{O}(T)$ and $n = \widetilde{O}(T)$; combining the state with a symbol is one Mul, i.e., $O(\log^3 q)$ products of ring elements of $O(n \log q)$ word operations each, and the lookup table costs $W - 1$ automorphisms, composed in $\log_2 W$ rounds, and W scalar multiplications for the lookup width $W = w|\Sigma|$; the factors $\log W \le \log n$ are absorbed by $\widetilde{O}$.

Table 3: Asymptotic comparison of our direct evaluation with representative leveled FHE-over-encoding constructions for a branching program of length T and width w over an alphabet Σ. In the FHE-based constructions, decryption represents the FHE evaluation of the branching program as a Boolean circuit and evaluates this circuit over BGG+ encodings; their costs include this evaluation. We assume $T = \Theta(\ell)$ for the attribute length ℓ, so that the terms in ℓ are written in T; our scheme requires the padded lookup width $W = w|\Sigma|$ to satisfy $W < n$. Dec. is the cost of decryption in word operations; we compare decryption because it is performed online for every ciphertext, whereas key generation is a preprocessing step performed once per predicate. $p := \mathrm{poly}(\lambda, \log T)$, and Q is the bound on authorized keys. For our scheme in this table and its derivation below, $\widetilde{O}$ suppresses polynomial factors in the security parameter λ as well as polylogarithmic factors in T, w, and $|\Sigma|$.

Scheme	$\log q$	$\lvert\mathsf{mpk}\rvert$	$\lvert\mathsf{sk}_F\rvert$	$\lvert\mathsf{ct}\rvert$	Dec.
[GVW15]	$\mathrm{poly}(\lambda, \log T)$	$T \log\lvert\Sigma\rvert \cdot p$	p	$T \log\lvert\Sigma\rvert \cdot p$	$T(w^3 + w\lvert\Sigma\rvert) \cdot p$
[BTVW17]	$\mathrm{poly}(\lambda, \log T)$	$T \log\lvert\Sigma\rvert \cdot p$	p	$T \log\lvert\Sigma\rvert \cdot p$	$T(w^3 + w\lvert\Sigma\rvert) \cdot p$
[WWLW25]	$\mathrm{poly}(\lambda, \log T)$	$(Q + T \log\lvert\Sigma\rvert) \cdot p$	p	$(Q + T \log\lvert\Sigma\rvert) \cdot p$	$T(w^3 + w\lvert\Sigma\rvert) \cdot p$
This work	$\widetilde{O}(T)$	$T \cdot \widetilde{O}(T^5)$	$w\lvert\Sigma\rvert \cdot \widetilde{O}(T^3)$	$T \cdot \widetilde{O}(T^5)$	$w\lvert\Sigma\rvert \cdot \widetilde{O}(T^6)$

of our scheme costs $O(w|\Sigma|)$ operations over encodings, compared with $O(w^3 + w|\Sigma|)$ gates for FHE-based schemes. Conversely, asymmetric multiplication forces step-by-step evaluation, so that our modulus bit size grows linearly in T up to logarithmic factors, whereas FHE-based schemes evaluate a balanced tree and their modulus bit size grows only polylogarithmically in T.

These asymptotics, however, hide the dominant cost of FHE-based schemes in the factor p: simulating FHE evaluation as a Boolean circuit increases the number of encoding operations, and the resulting depth requires a larger modulus and ring dimension. Table 1 quantifies this cost against a single simulated FHE multiplication.

PE via lockable obfuscation and ABE. A black-box construction of PE from ABE and lockable obfuscation is known [GKW17, WZ17]: a ciphertext includes an obfuscation, under a random lock, of the decryption of a hidden ABE ciphertext of the lock, so that the obfuscated program reveals the message exactly to the keys that decrypt the ABE ciphertext. Existing lockable obfuscation for general circuits, however, evaluates the obfuscated circuit homomorphically under FHE, and this construction is therefore not FHE-free either.[2]

Lattice-based ABE for circuits of a-priori bounded polynomial depth was first constructed by Gorbunov, Vaikuntanathan, and Wee [GVW13] and by Boneh et al. [BGG+14]; the latter introduced key-homomorphic encodings, and lattice-based key-policy ABE has been implemented by Dai et al. [DDP+18]. Evasive LWE [Wee22, Tsa22] leads to further ABE and PE constructions; in particular, Hsieh, Lin, and Luo [HLL23] support circuits of unbounded depth under evasive circular LWE by bootstrapping BGG+ encodings: an intermediate encoding is rounded and then refreshed by a homomorphic evaluation over a BGG+ encoding of a circular GSW encryption of the secret.

Homomorphic computation without FHE. Boyle, Gilboa, and Ishai [BGI16] use restricted-multiplication straight-line programs for homomorphic secret sharing, where, as here, a memory value can be multiplied only by an input. Boyle, Kohl, and Scholl [BKS19] realize such programs from lattices without FHE by performing each restricted multiplication as a distributed decryption.

[2] FHE-free lockable obfuscation is known for width-5 permutation branching programs [GKW17, WZ17]. Converting an ABE decryption circuit of depth d by Barrington's theorem [Bar89], however, yields a program of length $O(4^d)$, and the bit length of the modulus of the GGH15-style encodings grows linearly in the length of the program, and hence exponentially in d. We therefore exclude this route from the candidates for a practical construction.

Our asymmetric multiplication follows the same principle, decrypting a ciphertext of the input inside an encoding. In their scheme, however, rounding the shares refreshes the noise, so that the parameters do not depend on the multiplicative depth, whereas ours do.

AI Use Methodology

Activity	**AI used**
Asking the research question	[No]
Coming up with the approach	[No]
Proof development	[Yes]
Writing and exposition	[Yes]
Checking for bugs	[Yes]
Other supporting tasks	[Yes]

Details. The human authors used two AI models, GPT-6 Astra and Claude Opus 5.5, during the development of the research idea, the preparation of the manuscript, and the implementation, as described below. The human authors reviewed and revised all of the resulting constructions, proofs, figures, tables, and manuscript, and take full responsibility for this paper and its contents.

Origin of the research idea. The research initially aimed at making computation over private values on key-homomorphic encodings practically implementable in lattice-based advanced encryption and indistinguishability obfuscation (iO), not only in PE, and PE was considered as one of its concrete applications. To make such computation efficient, the human authors were investigating two concrete approaches in parallel.

The first approach encodes a private value directly in BGG+ encodings [BGG+14] as a one-hot vector, without FHE. The human authors focused on this approach because, when one of two inputs is given explicitly, multiplying an encoding by a gadget ciphertext yields an encoding of the one-hot vector that combines the two values, and a linear combination of encodings evaluates a lookup table (LUT) on a private input. They recognized, however, the growth of the encoding size as a problem of this approach. The second approach applies the SIMD packing proposed by Kim, Li, Lin, and Liu [KLLL26] for homomorphic secret sharing (HSS) to BGG+ encodings, so that a single encoding multiplies as many values as possible.

Building on both ideas, the AI models proposed encoding a value in the exponent of a single monomial instead of a one-hot vector. They further proposed applying the technique of [KLLL26] for clearing all coefficients of a ring element except those at the desired positions to BGG+ encodings, so as to evaluate a LUT over this exponent representation.

Construction and security proof. Using the encoding proposed by the AI models, the human authors designed the concrete PE construction and outlined its security proof. The AI models, however, found that the proof was incomplete in simulating preimages without a lattice trapdoor. They then completed the proof and made the necessary modifications to the construction, namely, using the automorphic images of the public matrices $\mathbf{B}$ and $\mathbf{H}_k$ in the key-generation algorithm.

Manuscript. For each part of the manuscript, the human authors provided an outline, from which the AI models wrote the first draft. The human authors then reviewed and revised the draft,

with the assistance of the AI models. Figures 1, 2, and 3 were created by the AI models following the human authors' instructions. The human authors then reviewed and revised them, with the assistance of the AI models. The human authors also used the AI models to survey the literature. Tables 2 and 3 were created by the AI models. The human authors checked the cited papers and verified that the contents of the tables, as well as their classification and analysis of the cited works, are correct.

Implementation and evaluation. The human authors specified the implementation at a high level, including its data structures, API design, and tests, and the AI models wrote all of the code of the implementation and the evaluation. The reported numbers were not measured by the AI models in an ad hoc manner: measurement code was added to the implementation, and the AI models read the numbers from the logs of its execution.

2 Technical Overview

2.1 Asymmetric Multiplication

BGG+ encodings and PE. We first recall BGG+ encodings [BGG+14], which we describe over the polynomial ring $\mathcal{R}_q := \mathbb{Z}_q[X]/(X^n+1)$ for a power of two n, as in our scheme, and an overview of the construction of PE from them in the line of work initiated by Gorbunov, Vaikuntanathan, and Wee [GVW15]. Let $\mathbf{s} = (\bar{s}, 1)$ for a secret $\bar{s} \in \mathcal{R}_q$, and let $\mathbf{G}$ be the gadget matrix. An encoding of an attribute $\mu \in \mathcal{R}_q$ under a public matrix $\mathbf{A}$ is

$$\mathbf{c} = \mathbf{s}(\mathbf{A} - \mu\mathbf{G}) + \mathbf{e},$$

which we write as $\mathbf{c} : [\![\mu\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}}$. Encodings are key-homomorphic: from encodings of μ_i under $\mathbf{A}_i$, anyone can compute an encoding of $F(\mu_1, \dots)$ under a matrix $\mathbf{A}_F$ that depends only on F and the $\mathbf{A}_i$. In PE, the ciphertext contains $\mathbf{b} = \mathbf{s}\mathbf{B} + \mathbf{e}_b$, encodings of the attribute, and $c_{\mathsf{m}} = \mathbf{s}\mathbf{d}^\top + e + \lfloor q/2 \rceil m$. A key for F is a short vector, called a preimage, $\mathbf{k} = (\mathbf{k}_1, \mathbf{k}_2)$ satisfying $[\mathbf{B} \mid \mathbf{A}_F]\mathbf{k}^\top = \mathbf{d}^\top$. The decryptor evaluates an encoding $\mathbf{c}_F$ of $F(x)$ and obtains

$$[\mathbf{b} \mid \mathbf{c}_F]\,\mathbf{k}^\top = \mathbf{s}\mathbf{d}^\top - F(x)\,\mathbf{s}\mathbf{G}\mathbf{k}_2^\top + \text{small},$$

which equals $\mathbf{s}\mathbf{d}^\top$ up to small noise exactly when $F(x) = 0$ and then unmasks m. Constructing PE thus reduces to computing an encoding of $F(x)$ from encodings of the hidden attribute x, under a public matrix that key generation can compute from the public matrices alone.

The obstacle: multiplying hidden values. Encodings directly support additions and multiplications by public ring elements. They also support the multiplication of two encodings $\mathbf{c}_1 : [\![\mu_1\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}_1}$ and $\mathbf{c}_2 : [\![\mu_2\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}_2}$ by the rule

$$\mathbf{c}_1\mathbf{G}^{-1}(\mathbf{A}_2) + \mu_1\mathbf{c}_2 : [\![\mu_1\mu_2\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}_1\mathbf{G}^{-1}(\mathbf{A}_2)}.$$

This rule requires the evaluator to know μ_1, but μ_2 may remain hidden. In ABE, where the attribute is public, this requirement is harmless; in PE, it rules out every nonlinear predicate of the hidden attribute, unless one factor of every multiplication is public. For general circuits, both factors of a multiplication may be intermediate values. Prior PE [GVW15, BTVW17] therefore makes every multiplied value public: it encrypts the attribute under FHE and encodes the bits of the FHE ciphertext, so that every multiplication in the circuit of FHE evaluation has public operands, but the predicate is then evaluated only through this circuit, gate by gate over encodings.

Keeping the state hidden. A branching program does not need every multiplied value to be public. Each step multiplies the current state by a value derived from a single attribute symbol, and the product, the new state, is used only as the state of the next step. The state can therefore play the role of the hidden factor μ_2 in every multiplication, as long as the factor derived from the symbol has a public form that can play the role of μ_1. As in prior PE, we give the symbol such a form as an encrypted value: the encryptor supplies a gadget ciphertext of the symbol, whose digits are public. What differs from prior PE is that the state remains hidden in its encoding, and that the digits are multiplied by this encoding instead of being processed by an evaluation of FHE.

Decryption inside an encoding. The multiplication of the state by the digits of a ciphertext decrypts the ciphertext inside the encoding. In PE schemes that evaluate FHE over BGG+ encodings, the dual-use technique of Brakerski et al. [BTVW17] lets the FHE secret key coincide with the secret of the encodings, so that FHE decryption, which is linear in the ciphertext up to the final rounding, happens inside the encodings at the end of the evaluation. We observe that such a decryption can multiply: decrypting a ciphertext of ν inside an encoding of μ yields an encoding of $\mu\nu$. Instead of evaluating FHE and decrypting once at the end, we therefore decrypt a fresh ciphertext of the symbol at every multiplication.

Let $\mathbf{c} : [\![\mu\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}}$ be an encoded state and ν an attribute symbol. The encryptor supplies a ciphertext $\mathbf{C} \in \mathcal{R}_q^{2\times 2\tau}$ of ν under the secret of the encodings, in the gadget form of GSW [GSW13],

$$\mathbf{sC} = \nu\mathbf{sG} + \mathbf{e}_{\mathbf{C}},$$

together with encodings $\mathbf{c}_{r,j,i} : [\![d_{r,j,i}\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}_{r,j,i}}$ of the binary digits $d_{r,j,i}$ of its entries, $C_{r,j} = \sum_i 2^i d_{r,j,i}$, under public matrices fixed at setup. As in the dual-use technique, $\mathbf{C}$ plays two roles. Since its digits are public, the evaluator can multiply every digit encoding by $\mathbf{c}$ with the rule above, with the public digit $d_{r,j,i}$ as μ_1 and the hidden state μ as μ_2, which yields an encoding of $d_{r,j,i}\mu$. Summing these with the weights 2^i reassembles the entries of $\mathbf{C}$, column by column:

$$\mathbf{c}'_j := \sum_{r,i} \big(\mathbf{c}_{r,j,i}\mathbf{G}^{-1}(\mathbf{A}) + d_{r,j,i}\mathbf{c}\big)\mathbf{G}^{-1}(2^i\mathbf{1}_r^\top) = \mathbf{sA}'_j - \mu\,(\mathbf{sC})_j + \mathbf{e}'_j.$$

At the same time, the secret key of $\mathbf{C}$ is the secret $\mathbf{s}$ of the encodings, and the payload $\mu\,\mathbf{sC}$ is a decryption of $\mathbf{C}$ multiplied by μ. Since $\mathbf{sC} = \nu\mathbf{sG} + \mathbf{e}_{\mathbf{C}}$, the row $\mathbf{c}' = (\mathbf{c}'_j)_j$ is an encoding of $\mu\nu$ under a public matrix $\mathbf{A}'$ that depends only on $\mathbf{A}$ and the $\mathbf{A}_{r,j,i}$, so that key generation can compute it. In summary, $\mathbf{C}$ is decrypted inside the encoding: ν is never revealed, $\mathbf{C}$ itself is never operated on homomorphically, and only the small term $\mu\mathbf{e}_{\mathbf{C}}$ joins the noise when μ has small norm. As in the homomorphic secret sharing of Boyle, Kohl, and Scholl [BKS19], where a distributed decryption multiplies an input by a memory value, a multiplication costs about one decryption instead of an FHE multiplication.

2.2 Lookup Tables over Exponents

No multiplication of the state. Since asymmetric multiplication multiplies the state only by fresh gadget ciphertexts of attribute symbols, two state values are never multiplied, and a step can apply to the state only operations that never multiply it by itself: additions, multiplications by public ring elements, and, as we show below, automorphisms. A transition of a branching program, however, is an arbitrary nonlinear function. We thus look for a representation of the state in which one encoding holds the whole state and arbitrary nonlinear functions are computable by such operations. Neither a single coefficient nor a binary representation of the state suffices,

since computing a nonlinear function of either requires multiplying the state by itself, which the restriction forbids.

Arbitrary nonlinear functions are computable if the state is stored as a one-hot vector. For values a and b, let $[a = b]$ denote 1 if $a = b$ and 0 otherwise. The state $z \in [0, w-1]$ is then stored as w encodings, one of $[z = z']$ for every $z' \in [0, w-1]$, so that exactly the encoding for $z' = z$ encodes 1; likewise, every attribute symbol $x \in \Sigma$ is supplied as $|\Sigma|$ gadget ciphertexts, one of $[x = x']$ for every $x' \in \Sigma$. The next state is then

$$[f(z, x) = z''] = \sum_{(z',x'):\, f(z',x')=z''} [z = z'] \cdot [x = x'],$$

which uses only products of state entries with supplied indicators. This approach, however, needs w encodings for the state and $|\Sigma|$ gadget ciphertexts for every attribute symbol, and the ciphertext thus grows by a factor of $|\Sigma|$, which is 256 for a byte alphabet. We instead propose a representation that keeps arbitrary nonlinear functions computable under the restriction, while representing a state among w values by a single encoding and an attribute symbol in Σ by a single gadget ciphertext. To this end, we view a nonlinear function on a finite domain as a lookup table and evaluate the table on the exponent of a monomial.

Exponent representation. Inspired by the programmable bootstrapping of TFHE [CGGI16, CJP21], which evaluates a lookup table on a value stored in the exponent of a monomial, we represent a value $u \in [0, W-1]$ by the monomial X^u, for a power of two $W \mid n$ with $W < n$, and evaluate tables on the exponent. The principle is as follows. Let $L : [0, W-1] \to [0, W-1]$ be a table. First, for each input $j \in [0, W-1]$ of the table, multiplying X^u by the public monomial X^{-j} gives X^{u-j}, whose constant term is 1 if $u = j$ and 0 otherwise, since $0 < |u - j| < W < n$ when $u \neq j$. Second, suppose that an operation π clears all coefficients except the constant term, so that $\pi(X^{-j}X^u) = [u = j]$. Multiplying it by the output monomial $X^{L(j)}$ and summing over all j then selects the row of the table that matches u:

$$X^{L(u)} = \sum_{j=0}^{W-1} X^{L(j)}\, \pi\big(X^{-j}X^u\big).$$

The multiplications by X^{-j} and $X^{L(j)}$ are multiplications by public ring elements. Hence, if π is realized by operations that never multiply the state by itself, then so is the whole table. Moreover, as long as the table size W is sufficiently small compared with the ring dimension n, an encoding of a single ring element suffices to hold the whole state, and a single gadget ciphertext suffices to hold a symbol.

The technique above easily extends to a table with two inputs, the state and an attribute symbol, as required for a transition $f(z, x)$ of a branching program. The idea is to extend the exponent of the encoded state from X^z to X^{z+wx} by an asymmetric multiplication, and then to evaluate as above the single-input table $L : [0, W-1] \to [0, W-1]$ with $L(z + wx) = f(z, x)$, where $W = w|\Sigma|$. Concretely, the encryptor supplies the symbol x as $\nu = X^{wx}$; since a product of monomials adds exponents, multiplying the encoded state X^z by ν gives X^{z+wx}, whose exponent $z + wx \in [0, W-1]$ determines the pair (z, x) uniquely.

Lookup tables via automorphisms. It remains to realize π. For this, we borrow the idea of Kim, Li, Lin, and Liu [KLLL26] for clearing all coefficients of a ring element except those at the desired positions, and combine it with the exponent representation to select the row of a table. We first explain it on plain ring elements and then turn to encodings.

position	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$	$p=7$
σ_1 $(j=0)$	1	1	1	1	1	1	1	1
σ_9 $(j=1)$	1	X^8	-1	$-X^8$	1	X^8	-1	$-X^8$
σ_{17} $(j=2)$	1	-1	1	-1	1	-1	1	-1
σ_{25} $(j=3)$	1	$-X^8$	-1	X^8	1	$-X^8$	-1	X^8
sum	4	0	0	0	4	0	0	0

Figure 1: The coefficient sieve for $W = 4$ and $n = 16$, where $\mathcal{H} = \{1, 9, 17, 25\}$ and $X^{16} = -1$. For $k = 1 + 8j$, the automorphism σ_k maps X^p to $X^{8jp} \cdot X^p$; the entry in row σ_k and column p is the factor X^{8jp}, and the last row is the sum over $k \in \mathcal{H}$. In the shaded columns, where $4 \mid p$, every factor equals 1 and the sum is 4; in the other columns, the factors cancel, as $1 + X^8 - 1 - X^8 = 0$ or $1 - 1 + 1 - 1 = 0$. Hence $\sum_{k\in\mathcal{H}} \sigma_k$ keeps exactly the coefficients at positions divisible by W, scaled by W; the pattern repeats with period 4 in p.

We need not clear every nonconstant coefficient. Let π_W keep the coefficients at positions divisible by W and set all others to zero. Since $u - j \in (-W, W)$, $W \mid n$, and $X^{-a} = -X^{n-a}$, the monomial X^{u-j} sits at position 0 if $u = j$ and at a position not divisible by W otherwise; hence $\pi_W(X^{u-j}) = [u = j]$, and π_W can replace π in the selection formula.

The projection π_W can be realized as an average of ring automorphisms. For an odd k, let σ_k be the ring automorphism that maps X to X^k, and hence each monomial X^p to $X^{k\cdot p}$. For a power of two $W \mid n$ with $W < n$, define

$$\mathcal{H} := \{1 + j \cdot 2n/W : j \in [0, W - 1]\} \subseteq \mathbb{Z}^*_{2n}.$$

The elements of $\mathcal{H}$ are automorphism indices, with multiplication modulo $2n$ corresponding to composition of automorphisms. This group has W elements, chosen so that

$$\pi_W(z) = W^{-1} \sum_{k\in\mathcal{H}} \sigma_k(z).$$

Here W^{-1} is taken modulo q; it exists because q is odd. The following examples explain why this average keeps exactly the coefficients at positions divisible by W. Consider first $W = 2$, for which $\mathcal{H} = \{1, n + 1\}$. Since $X^n = -1$, the automorphism σ_{n+1} maps X to $X^{n+1} = -X$, and hence X^p to $(-1)^p X^p$. The sum $z + \sigma_{n+1}(z)$ therefore doubles the coefficients of z at even positions and cancels those at odd positions, that is, it equals $2\,\pi_2(z)$. For $W = 4$, we have $\mathcal{H} = \{1, n/2+1, n+1, 3n/2+1\}$ and additionally use $\sigma_{n/2+1}$, which maps X to $X^{n/2} \cdot X$, and hence X^p to $X^{pn/2} \cdot X^p$, where $X^{pn/2}$ equals 1, $X^{n/2}$, -1, or $-X^{n/2}$ according as $p \equiv 0, 1, 2, 3 \pmod 4$. Summing the images $\sigma_k(X^p)$ over $k \in \mathcal{H}$ gives $X^p \sum_{j=0}^{3} X^{jpn/2}$, so it multiplies X^p by $1 + X^{pn/2} + X^{pn} + X^{3pn/2}$, which equals 4 if $4 \mid p$ and vanishes otherwise. Figure 1 illustrates this cancellation for $n = 16$: each row lists the factors by which one automorphism multiplies the monomials $X^0, \ldots, X^7$, and only the columns with $4 \mid p$ add up to a nonzero value. In general, the same cancellation gives $\sum_{k\in\mathcal{H}} \sigma_k(z) = W \cdot \pi_W(z)$; division by W gives the desired projection (Lemma 7).

Substituting this into the selection formula and moving the public factors out of the automorphisms gives

$$X^{L(u)} = \sum_{k\in\mathcal{H}} \alpha_{k,L}\, \sigma_k(X^u), \qquad \alpha_{k,L} := W^{-1} \sum_{j=0}^{W-1} X^{L(j)} \sigma_k(X^{-j}),$$

where the $\alpha_{k,L}$ are public ring elements determined by the table. Since $\sigma_k(X^u) = (X^u)^k$, this is a polynomial in the state with public coefficients, whose powers are computed by automorphisms instead of multiplications of the state by itself.

Automorphisms over encodings. The table is thus evaluated by automorphisms and multiplications by public ring elements alone. The latter are available on encodings, and it remains to evaluate automorphisms on encodings. The difficulty is that an automorphism also acts on the secret, as in FHE, where it is resolved by key switching. Since σ_k is a ring automorphism that fixes $\mathbf{G}$, applying it entrywise to an encoding gives

$$\sigma_k(\mathbf{c}) = \sigma_k(\mathbf{s})\big(\sigma_k(\mathbf{A}) - \sigma_k(\mu)\mathbf{G}\big) + \sigma_k(\mathbf{e}) : [\![\sigma_k(\mu)\mathbf{s}_k]\!]_{\mathbf{s}_k}^{\sigma_k(\mathbf{A})}, \qquad \mathbf{s}_k := \sigma_k(\mathbf{s}) = (\sigma_k(\bar{s}), 1).$$

The attribute is now $\sigma_k(\mu)$, as desired, and the public matrix $\sigma_k(\mathbf{A})$ is public, but both the mask and the payload have moved to the conjugate secret $\mathbf{s}_k$. We move them back in two steps.

First, we switch the payload. The payload $\sigma_k(\mu)\mathbf{s}_k\mathbf{G}$ has exactly the form required for an asymmetric multiplication under $\mathbf{s}_k$; multiplying by a gadget ciphertext $\mathbf{C}_k$ with $\mathbf{s}_k\mathbf{C}_k \approx \mathbf{s}\mathbf{G}$, through encodings of its digits under $\mathbf{s}_k$, replaces it with $\sigma_k(\mu)\mathbf{s}\mathbf{G}$; the mask is still $\mathbf{s}_k$. Second, we switch the mask with a helper $\mathbf{h}_k = \mathbf{s}\mathbf{H}_k - \mathbf{s}_k\mathbf{G} + \mathbf{e}_k$, which is an encoding of $\mathbf{s}_k$ under $\mathbf{s}$ and plays the role of a key-switching key in FHE: for any row $\mathbf{c}' = \mathbf{s}_k\mathbf{A}' - \mathbf{p} + \mathbf{e}'$,

$$\mathbf{h}_k\mathbf{G}^{-1}(\mathbf{A}') + \mathbf{c}' = \mathbf{s}\,\mathbf{H}_k\mathbf{G}^{-1}(\mathbf{A}') - \mathbf{p} + \text{small},$$

since the payload $\mathbf{s}_k\mathbf{G}\mathbf{G}^{-1}(\mathbf{A}') = \mathbf{s}_k\mathbf{A}'$ of the first term cancels the mask of the second. The result is an encoding of $\sigma_k(\mu)$ under $\mathbf{s}$, with a public matrix computed from $\sigma_k(\mathbf{A})$, $\mathbf{H}_k$, and the public matrices of the digit encodings only, so that key generation can follow the same steps.

The gadget ciphertext $\mathbf{C}_k$ itself need not be sent, since anyone can compute it from the helpers: applying σ_k to $\mathbf{h}_{k^{-1}}$ gives $\sigma_k(\mathbf{h}_{k^{-1}}) = \mathbf{s}_k\sigma_k(\mathbf{H}_{k^{-1}}) - \mathbf{s}\mathbf{G} + \text{small}$, and since the second entry of $\mathbf{s}_k$ is 1, the matrix $\mathbf{C}_k := \sigma_k(\mathbf{H}_{k^{-1}}) - \binom{\mathbf{0}}{\sigma_k(\mathbf{h}_{k^{-1}})}$ satisfies $\mathbf{s}_k\mathbf{C}_k \approx \mathbf{s}\mathbf{G}$. The encodings of its digits, however, must be part of the ciphertext: as for any gadget ciphertext of the encryptor, the multiplication goes through these digit encodings, under public matrices fixed at setup, so that the resulting public matrix does not depend on $\mathbf{C}_k$, which varies with each ciphertext. The ciphertext therefore contains a helper and the digit encodings of $\mathbf{C}_k$ for every automorphism σ_k that the evaluator applies, and the helper for k^{-1} from which $\mathbf{C}_k$ is computed. As we show below, the evaluator applies only $O(\log W)$ distinct automorphisms, so that the ciphertext contains only $O(\log W)$ helpers and sets of digit encodings.

LUT evaluation over encodings. Putting everything together, we obtain a lookup-table evaluation over encodings. Combining both switches yields an operation Aut_k that maps an encoding of μ to an encoding of $\sigma_k(\mu)$ under the same secret. Given an encoding $\mathbf{c}$ of X^u, the evaluator computes encodings of $\sigma_k(X^u)$ for all $k \in \mathcal{H}$, multiplies them by the public ring elements $\alpha_{k,L}$, and sums them, which yields an encoding of $X^{L(u)}$ under a public matrix that key generation computes in the same way.

Applying each Aut_k to $\mathbf{c}$ directly would require a helper and the digit encodings of $\mathbf{C}_k$ for every $k \in \mathcal{H} \setminus \{1\}$, i.e., $W - 1$ gadget ciphertexts in every ciphertext. Instead, the evaluator composes automorphisms along a binary basis $g_0, \ldots, g_{b-1}$ of $\mathcal{H}$, where $b = \log_2 W$: in round j, it applies Aut_{g_j} to every encoding obtained so far, which doubles their number, and after b rounds it holds encodings of $\sigma_k(X^u)$ for all $k \in \mathcal{H}$. Since Aut_{g_j} returns an encoding under the same secret $\mathbf{s}$, it can be applied again in later rounds with the same helper data; composing automorphisms therefore requires no helper data for the composite σ_k. The ciphertext thus contains only b gadget ciphertexts $\mathbf{C}_{g_j}$ and at most $2b$ helpers, at the cost of a table of depth b.

Each step of a branching program thus consists of one asymmetric multiplication and one such table, i.e., $W - 1$ automorphisms in b rounds and W multiplications by public ring elements, all on a single encoding (Figure 2).

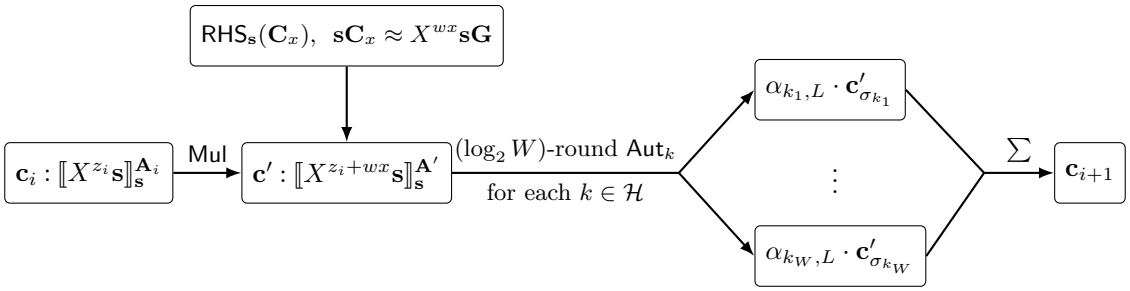

Figure 2: One step $z_{i+1} = f_i(z_i, x)$ of a branching program over encodings. The asymmetric multiplication Mul combines the encoding $\mathbf{c}_i$ of the state X^{z_i} with the RHS data of the gadget ciphertext $\mathbf{C}_x$ of the symbol X^{wx}, and yields an encoding $\mathbf{c}'$ of X^{z_i+wx}. The lookup table L with $L(z+wx) = f_i(z,x)$ is then evaluated as a linear combination, with public coefficients $\alpha_{k,L}$, of the encodings $\mathbf{c}'_{\sigma_k}$ of $\sigma_k(X^{z_i+wx})$ for $k \in \mathcal{H} = \{k_1, \ldots, k_W\}$, which yields an encoding $\mathbf{c}_{i+1}$ of $X^{z_{i+1}}$. The encodings $\mathbf{c}'_{\sigma_k}$ are computed from $\mathbf{c}'$ by the automorphisms Aut_{g_j} along a binary basis $g_0, \ldots, g_{b-1}$ of $\mathcal{H}$, in $b = \log_2 W$ rounds.

3 Preliminaries

3.1 Notations

Let λ be the security parameter. A function $f : \mathbb{N} \to [0, \infty)$ is negligible, written $f(\lambda) = \mathrm{negl}(\lambda)$, if for every constant $c > 0$ there exists $\lambda_0 \in \mathbb{N}$ such that $f(\lambda) < \lambda^{-c}$ for all $\lambda > \lambda_0$. PPT stands for probabilistic polynomial time. The statistical distance between distributions X and Y over a countable set is $\Delta(X, Y) := \frac{1}{2}\sum_a |\Pr[X = a] - \Pr[Y = a]|$. For distribution ensembles $X = \{X_\lambda\}$ and $Y = \{Y_\lambda\}$, we write $X \approx_s Y$ if $\Delta(X_\lambda, Y_\lambda) = \mathrm{negl}(\lambda)$, and $X \approx_c Y$ if every PPT algorithm distinguishes them with advantage $\mathrm{negl}(\lambda)$. We write $x \leftarrow\!\$\ D$ for sampling x from a distribution D, and $x \leftarrow\!\$\ S$ for sampling x uniformly from a finite set S. $\widetilde{O}(\cdot)$ hides polylogarithmic factors.

For integers $i \le j$, $[i, j] := \{i, i+1, \ldots, j\}$, and $[n] := [1, n]$. Vectors and matrices are written in bold lowercase and uppercase letters, respectively, and vectors are row vectors unless stated otherwise. $\mathbf{I}_k$ is the $k \times k$ identity matrix and $\otimes$ the Kronecker product. For a matrix $\mathbf{M}$ with two rows, $\mathbf{M}^{(1)}$ and $\mathbf{M}^{(2)}$ denote its first and second rows. $\lfloor \cdot \rceil$ denotes rounding to the nearest integer.

Let n be a power of two, $\mathcal{R} := \mathbb{Z}[X]/(X^n + 1)$, and $\mathcal{R}_q := \mathcal{R}/q\mathcal{R}$ for an odd modulus q, whose elements are represented by polynomials of degree less than n with coefficients in $(-q/2, q/2]$. For $a \in \mathcal{R}$, $\|a\|_\infty$ is the largest absolute value of its coefficients and $\|a\|_1$ their sum; for a vector or matrix over $\mathcal{R}$, $\|\cdot\|_\infty$ is the maximum over its entries. For $\alpha \in \mathcal{R}$, a row $\mathbf{e}$, and a matrix $\mathbf{M}$ with k rows,

$$\|\alpha\mathbf{e}\|_\infty \le \|\alpha\|_1 \|\mathbf{e}\|_\infty, \qquad \|\mathbf{e}\mathbf{M}\|_\infty \le k \cdot \max_{i,j} \|M_{i,j}\|_1 \cdot \|\mathbf{e}\|_\infty.$$

3.2 Lattice Preliminaries

Gadget matrix. Let $\tau := \lceil \log_2 q \rceil$, $\mathbf{g} := (1, 2, \ldots, 2^{\tau-1})$, and $\mathbf{G} := \mathbf{I}_2 \otimes \mathbf{g} \in \mathcal{R}_q^{2 \times 2\tau}$, so that $\mathbf{G}^{(1)} = (\mathbf{g}, \mathbf{0})$ and $\mathbf{G}^{(2)} = (\mathbf{0}, \mathbf{g})$. For $\mathbf{M} \in \mathcal{R}_q^{2 \times a}$, $\mathbf{G}^{-1}(\mathbf{M}) \in \mathcal{R}^{2\tau \times a}$ denotes the coefficient-wise binary decomposition, so that $\mathbf{G}\mathbf{G}^{-1}(\mathbf{M}) = \mathbf{M}$ and $\|\mathbf{e}\mathbf{G}^{-1}(\mathbf{M})\|_\infty \le 2\tau n \|\mathbf{e}\|_\infty$. We write $\mathbf{1}_r \in \{0,1\}^{1 \times 2}$ for the r-th unit vector.

Automorphisms. For odd $k \in \mathbb{Z}_{2n}$, the ring automorphism $\sigma_k : X \mapsto X^k$ permutes coefficients up to sign, preserves norms, and satisfies $\sigma_k \circ \sigma_{k'} = \sigma_{kk'}$. It acts entrywise on vectors and matrices and fixes $\mathbf{G}$. All index products and inverses below are taken modulo $2n$.

Throughout, the lookup width W is a power of two satisfying $W \mid n$ and $W < n$ (hence $W \le n/2$). Define the order-W subgroup

$$\mathcal{H} := \{1 + j \cdot 2n/W : j \in [0, W-1]\} \subseteq \mathbb{Z}_{2n}^*.$$

These are the indices used in the lookup sum: averaging σ_k over $k \in \mathcal{H}$ keeps exactly the coefficients at positions divisible by W (Lemma 7). To obtain all these images by composition, fix a *binary basis* $g_0, \ldots, g_{b-1}$, where $b := \log_2 W$: every $k \in \mathcal{H}$ has a unique representation $k = \prod_{j<b} g_j^{e_j}$ with $e_j \in \{0,1\}$. Under $W < n$, the group is cyclic: $\mathcal{H} = \langle g \rangle$ for $g := 1 + 2n/W$, and we take $g_j := g^{2^j}$ for $j < b$. Let

$$\mathcal{K} := \{g_0, \ldots, g_{b-1}\}, \qquad \mathcal{J} := \mathcal{K} \cup \mathcal{K}^{-1}.$$

Thus $\mathcal{H}$ indexes all W required images, $\mathcal{K}$ the b automorphisms applied directly, and $\mathcal{J}$ the at most $2b$ helpers supplied in the ciphertext. Both k and k^{-1} are needed because Aut_k uses $\mathbf{h}_{k^{-1}}$ to switch the payload secret and $\mathbf{h}_k$ to switch the mask secret (Subsection 4.2).

For example, when $n = 16$ and $W = 4$,

$$\mathcal{H} = \{1, 9, 17, 25\}, \qquad \mathcal{K} = \{9, 17\}, \qquad \mathcal{J} = \{9, 17, 25\}.$$

The two evaluation rounds generate the indices $\{1\} \to \{1, 9\} \to \{1, 9, 17, 25\}$, since $9 \cdot 17 = 25 \pmod{32}$. The helper set also contains 25 because $9^{-1} = 25$, whereas $17^{-1} = 17$; no direct application of Aut_{25} is needed.

Matrices with one row per conjugate $\sigma_i(\bar{s})$ are indexed by $i \in \mathcal{H}$ in a fixed order starting with $i = 1$. Let $\boldsymbol{\delta}_i \in \{0,1\}^{1\times W}$ be the corresponding unit vector. For a row $\mathbf{a} \in \mathcal{R}_q^{1\times c}$, the matrix $\boldsymbol{\delta}_i^\top \otimes \mathbf{a}$ places $\mathbf{a}$ in row i and zeros elsewhere; for $\mathbf{v} \in \mathcal{R}_q^{1\times W}$, $\mathbf{v}\boldsymbol{\delta}_i^\top$ selects entry i.

Discrete Gaussians over $\mathcal{R}$. We identify $\mathcal{R}^N$ with the row vectors $\mathcal{R}^{1\times N}$ and view it as a lattice through the coefficient embedding, which maps $\mathbf{a} = (a_1, \ldots, a_N) \in \mathcal{R}^N$ to the concatenation of the coefficient vectors of $a_1, \ldots, a_N$ in $\mathbb{Z}^{nN}$. We write $\|\mathbf{a}\|_2$ for the Euclidean norm of the embedded vector; its largest absolute entry is $\|\mathbf{a}\|_\infty$ as defined in Subsection 3.1. For $\sigma > 0$, let $\rho_\sigma(\mathbf{a}) := \exp(-\pi\|\mathbf{a}\|_2^2/\sigma^2)$.

A *module lattice* is an $\mathcal{R}$-submodule $\Lambda \subseteq \mathcal{R}^N$ of full rank under the coefficient embedding. For a module lattice Λ, a shift $\mathbf{c} \in \mathcal{R}^N$, and $\sigma > 0$, the discrete Gaussian $D_{\Lambda+\mathbf{c},\sigma}$ is the distribution over $\Lambda + \mathbf{c} \subseteq \mathcal{R}^N$ that outputs $\mathbf{x}$ with probability proportional to $\rho_\sigma(\mathbf{x})$. The smoothing parameter $\eta_\epsilon(\Lambda)$ is that of Λ under the coefficient embedding [MR07].

We use two instances. First, $D_{\mathcal{R}^N,\sigma}$ outputs N ring elements whose nN coefficients are independent samples from $D_{\mathbb{Z},\sigma}$, and we write $D_{\mathcal{R},\sigma} := D_{\mathcal{R}^1,\sigma}$; here $\eta_\epsilon(\mathcal{R}^N) \le \omega(\sqrt{\log nN})$ for negligible ϵ. Second, for $\mathbf{M} \in \mathcal{R}_q^{h\times N}$ and $\mathbf{y} \in \mathcal{R}_q^{1\times h}$, the set

$$\Lambda_{\mathbf{y}}(\mathbf{M}) := \{\mathbf{k} \in \mathcal{R}^{1\times N} : \mathbf{M}\mathbf{k}^\top \equiv \mathbf{y}^\top \pmod q\}$$

is a coset of the module lattice $\Lambda^\perp(\mathbf{M}) := \Lambda_{\mathbf{0}}(\mathbf{M})$, and $D_{\Lambda_{\mathbf{y}}(\mathbf{M}),\sigma}$ is the discrete Gaussian over it.

All noise in this work and all secret keys are drawn from such distributions. Their size is bounded by the following tail bound.

Lemma 1 (Tail bound [Ban93, MP12])**.** *Let $\Lambda \subseteq \mathcal{R}^N$ be a module lattice with $nN = \mathrm{poly}(\lambda)$, $\mathbf{c} \in \mathcal{R}^N$, and $\sigma \ge \eta_\epsilon(\Lambda)$ for a negligible ϵ. Then*

$$\Pr_{\mathbf{x} \leftarrow\$ D_{\Lambda+\mathbf{c},\sigma}}\left[\|\mathbf{x}\|_\infty > \sigma\sqrt{\lambda}\right] \le \mathrm{negl}(\lambda).$$

We use this bound for $D_{\mathcal{R}^N,\sigma}$ with $\Lambda = \mathcal{R}^N$, and for $D_{\Lambda_{\mathbf{y}}(\mathbf{M}),\sigma}$ with $\Lambda = \Lambda^\perp(\mathbf{M})$.

Leftover hash lemma. The uniformity of trapdoors and the security proof rely on a leftover hash lemma over $\mathcal{R}_q$. The following lemma extends [JLWG25, Theorem 5.3], which is stated for a prime modulus, to a product of primes as in our scheme.

Lemma 2 (Leftover hash lemma over $\mathcal{R}_q$)**.** *Let $q = \prod_{i=1}^{L_q} q_i$ be a product of L_q distinct primes $q_i \equiv 1 \pmod{2n}$, let $q_{\min} := \min_i q_i$ and $\sigma_0 := \frac{\sqrt{\pi}}{2} \cdot \frac{q_{\min}}{\sqrt{\ln(4n)}}$, and let $k, m \geq 1$, $\epsilon \in (0, 1/2]$, $\omega(\sqrt{\log n}) \leq \sigma \leq \sigma_0$, and $\mathbf{r} \hookleftarrow\$ D_{\mathcal{R}^m,\sigma}$. If*

$$m(\log \sigma - 1) \;\geq\; k \log q + L_q \log(2n) + 2\log(1/\epsilon),$$

then for $\mathbf{A} \hookleftarrow\$ \mathcal{R}_q^{k\times m}$ and $\mathbf{u} \hookleftarrow\$ \mathcal{R}_q^k$,

$$\Delta\big((\mathbf{A}, \mathbf{A}\mathbf{r}^\top),\ (\mathbf{A}, \mathbf{u}^\top)\big) \leq \epsilon.$$

For a matrix $\mathbf{R}$ with c independent columns, each distributed as $\mathbf{r}^\top$, the statistical distance between $(\mathbf{A}, \mathbf{AR})$ and $(\mathbf{A}, \mathbf{U})$ for uniform $\mathbf{U}$ is at most $c\epsilon$.

Proof. Modulo each q_i, the polynomial X^n+1 has n distinct roots $\omega_{i,1}, \ldots, \omega_{i,n}$, and the prime ideals $\mathfrak{p}_{i,j} := q_i\mathcal{R} + (X - \omega_{i,j})\mathcal{R}$ satisfy $q\mathcal{R} = \prod_{i,j} \mathfrak{p}_{i,j}$; reduction modulo $\mathfrak{p}_{i,j}$ maps r to $r(\omega_{i,j}) \bmod q_i$, the entry of the NTT representation of r at the root $\omega_{i,j}$. For an ideal $\mathfrak{a} \supseteq q\mathcal{R}$, $N(\mathfrak{a}) := |\mathcal{R}/\mathfrak{a}|$ is its norm, and $\mathbf{r} \bmod \mathfrak{a}$ is taken entrywise for vectors $\mathbf{r}$. For a random variable X, $\mathrm{Col}(X) := \Pr[X = X']$ for an independent copy X' of X. We follow the proof of [JLWG25, Theorem 5.3] for a prime modulus. By [JLWG25, Lemma 5.1],

$$\Delta\big((\mathbf{A}, \mathbf{A}\mathbf{r}^\top),\ (\mathbf{A}, \mathbf{u}^\top)\big) \leq \frac{1}{2}\sqrt{\sum_{\mathfrak{a}\supseteq q\mathcal{R}} N(\mathfrak{a})^k \,\mathrm{Col}(\mathbf{r} \bmod \mathfrak{a}) - 1},$$

where the sum is over all ideals $\mathfrak{a} \supseteq q\mathcal{R}$, i.e., over all sets of NTT slots. The term for $\mathfrak{a} = \mathcal{R}$ equals 1 and cancels the -1, and it remains to bound the other terms.

Let $\mathfrak{a} \neq \mathcal{R}$, i.e., $\mathfrak{a}$ is divisible by at least one $\mathfrak{p}_{i,j}$. Then $\mathfrak{a} = \prod_i \mathfrak{a}_i$ with $\mathfrak{a}_i \supseteq q_i\mathcal{R}$, where $\mathfrak{a}_i$ is a product of t_i of the prime ideals $\mathfrak{p}_{i,j}$; let $t := \max_i t_i$, attained at i^*, so that $t \geq 1$. First, $N(\mathfrak{a}) = \prod_i q_i^{t_i} \leq \prod_i q_i^t = q^t$. Second, a collision modulo $\mathfrak{a}$ implies a collision modulo $\mathfrak{a}_{i^*}$, which is an ideal over the single prime q_{i^*} with norm $q_{i^*}^t$. Since $\sigma \leq \sigma_0 \leq \frac{\sqrt{\pi}}{2} \cdot \frac{q_{i^*}}{\sqrt{\ln(4t)}}$ by $q_{\min} \leq q_{i^*}$ and $t \leq n$, [JLWG25, Corollary 4] applies to $\mathfrak{a}_{i^*}$ and the prime q_{i^*}, and shows that each entry of $\mathbf{r} \bmod \mathfrak{a}_{i^*}$ has min-entropy at least $t\log\sigma - 1$, i.e., takes each value with probability at most $2\sigma^{-t}$. Since the collision probability of a random variable is at most the largest probability of a value, and since the m entries are independent, $\mathrm{Col}(\mathbf{r} \bmod \mathfrak{a}) \leq \mathrm{Col}(\mathbf{r} \bmod \mathfrak{a}_{i^*}) \leq (2\sigma^{-t})^m = 2^m\sigma^{-mt}$. Hence each term with $\mathfrak{a} \neq \mathcal{R}$ satisfies

$$N(\mathfrak{a})^k \,\mathrm{Col}(\mathbf{r} \bmod \mathfrak{a}) \leq 2^m\Big(\frac{q^k}{\sigma^m}\Big)^t \leq \theta^t, \qquad \theta := \frac{2^m q^k}{\sigma^m},$$

where q^t bounds the norm, which involves the slots of all primes, whereas the collision bound uses only the slots of q_{i^*}; the bound thus has the same form as that of [JLWG25, Theorem 5.3] for a prime modulus.

It remains to count the ideals. For each i, the number of choices of $\mathfrak{a}_i$ with $t_i \leq t$ is $\sum_{s\leq t}\binom{n}{s} \leq (n+1)^t \leq (2n)^t$, and hence at most $(2n)^{tL_q}$ ideals have the same t. The condition of the lemma is equivalent to $(2n)^{L_q}\theta \leq \epsilon^2$, and therefore

$$\sum_{\mathfrak{a}\neq\mathcal{R}} N(\mathfrak{a})^k \,\mathrm{Col}(\mathbf{r} \bmod \mathfrak{a}) \leq \sum_{t\geq 1}\big((2n)^{L_q}\theta\big)^t \leq \sum_{t\geq 1}\epsilon^{2t} \leq \frac{4}{3}\epsilon^2,$$

since $\epsilon \le 1/2$. The statistical distance is thus at most $\frac{1}{2}\sqrt{4\epsilon^2/3} < \epsilon$. The bound for a matrix $\mathbf{R}$ follows by a hybrid over its c columns, which replaces $\mathbf{A}$ times one column by a uniform column at a time. □

Smudging lemma. The security proof hides small terms by noise flooding, using the following coefficientwise extension of the scalar smudging lemma. We apply the following lemma entrywise to vectors with polynomially many entries, so that the total statistical distance remains negligible.

Lemma 3 (Smudging lemma [WWW22])**.** *Let $B, \sigma > 0$ with $\sigma \ge B \cdot \lambda^{\omega(1)}$. For every $a \in \mathcal{R}$ with $\|a\|_\infty \le B$,*

$$\Delta\big(D_{\mathcal{R},\sigma},\ a + D_{\mathcal{R},\sigma}\big) \le \mathrm{negl}(\lambda).$$

Trapdoors. Key generation uses trapdoor generation and preimage sampling, and extends the trapdoors to a larger matrix.

Lemma 4 (Trapdoor generation and preimage sampling [GPV08, MP12])**.** *For $h \ge 1$, let $\mathbf{G}_h := \mathbf{I}_h \otimes \mathbf{g} \in \mathcal{R}_q^{h \times h\tau}$. There are PPT algorithms* TrapGen *and* SamplePre *defined as follows.*

- $\mathsf{TrapGen}(1^n, q, h, \bar{m}, \sigma_{\mathbf{T}})$ *outputs $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G}_h - \bar{\mathbf{A}}\mathbf{T}] \in \mathcal{R}_q^{h \times (\bar{m}+h\tau)}$ and a trapdoor $\mathbf{T}$, where $\bar{\mathbf{A}}$ is uniform over $\mathcal{R}_q^{h\times\bar{m}}$ and $\mathbf{T}$ is distributed as $D_{\mathcal{R}^{\bar{m}\times h\tau},\sigma_{\mathbf{T}}}$.*
- $\mathsf{SamplePre}(\mathbf{A}, \mathbf{X}, \mathbf{u}, \sigma)$ *takes any $\mathbf{A} \in \mathcal{R}_q^{h\times m}$ and a* gadget trapdoor *$\mathbf{X} \in \mathcal{R}^{m\times h\tau}$ of $\mathbf{A}$, i.e., a matrix with $\mathbf{A}\mathbf{X} = \mathbf{G}_h$, a target $\mathbf{u} \in \mathcal{R}_q^{1\times h}$, and a width σ, and outputs $\mathbf{x} \in \mathcal{R}^{1\times m}$ with $\mathbf{A}\mathbf{x}^\top = \mathbf{u}^\top$. The output $\mathbf{A}$ of* TrapGen *has the gadget trapdoor $\binom{\mathbf{T}}{\mathbf{I}_{h\tau}}$, and we write $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \mathbf{u}, \sigma)$ for $\mathsf{SamplePre}\big(\mathbf{A}, \binom{\mathbf{T}}{\mathbf{I}_{h\tau}}, \mathbf{u}, \sigma\big)$.*

They satisfy the following properties.

- Uniformity (statistical instantiation): *if $\bar{m}$ and $\sigma_{\mathbf{T}}$ satisfy Lemma 2 for h rows, then $\mathbf{A}$ is within negligible statistical distance from uniform over $\mathcal{R}_q^{h\times(\bar{m}+h\tau)}$.*
- Preimage sampling: *if $\sigma \ge s_1(\mathbf{X}) \cdot \omega(\sqrt{\log n})$, where $s_1(\mathbf{X})$ is the largest singular value of $\mathbf{X}$ in the coefficient embedding, then for every $\mathbf{u} \in \mathcal{R}_q^{1\times h}$, the output of $\mathsf{SamplePre}(\mathbf{A}, \mathbf{X}, \mathbf{u}, \sigma)$ is within negligible statistical distance from $D_{\Lambda_{\mathbf{u}}(\mathbf{A}),\sigma}$. For the output of* TrapGen*, $s_1\big(\binom{\mathbf{T}}{\mathbf{I}_{h\tau}}\big) \le s_1(\mathbf{T}) + 1$, so that $\sigma \ge s_1(\mathbf{T}) \cdot \omega(\sqrt{\log n})$ suffices.*

The scheme uses TrapGen with $h = 1$, and we write $m_B := \bar{m} + \tau$ for the width of its output. Key generation extends these trapdoors to a larger matrix.

Lemma 5 (SampleLeft [ABB10, CHKP10])**.** *There is a PPT algorithm* SampleLeft *defined as follows.*

- $\mathsf{SampleLeft}(\mathbf{M}, \mathbf{T}, \mathbf{y}, \sigma)$ *takes a matrix $\mathbf{M} = [\mathbf{A} \mid \mathbf{M}_1] \in \mathcal{R}_q^{h\times N}$, where $\mathbf{A} \in \mathcal{R}_q^{h\times m}$ has the form of Lemma 4 with trapdoor $\mathbf{T}$, a target $\mathbf{y} \in \mathcal{R}_q^{1\times h}$, and a width σ, and outputs $\mathbf{x} \in \mathcal{R}^N$ with $\mathbf{M}\mathbf{x}^\top = \mathbf{y}^\top$.*

It satisfies the following property.

- Distribution: *if $\sigma \ge s_1(\mathbf{T}) \cdot \omega(\sqrt{\log n})$, then the output is within negligible statistical distance from $D_{\Lambda_{\mathbf{y}}(\mathbf{M}),\sigma}$.*

The key generation algorithm in Subsection 5.1 applies SampleLeft to a matrix whose trapdoor part consists of conjugates of a single row $\mathbf{a}$ output by TrapGen.

Lemma 6 (SampleLeftBlock)**.** *Let $(\mathbf{a}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(1^n, q, 1, \bar{m}, \sigma_{\mathbf{T}})$, so that $\mathbf{a} \in \mathcal{R}_q^{1\times m_B}$. There is a PPT algorithm* SampleLeftBlock *defined as follows.*

- $\mathsf{SampleLeftBlock}(\mathbf{M}, \mathbf{T}, \mathbf{y}, \sigma)$ *takes a matrix* $\mathbf{M} = [\mathbf{M}_0 \mid \mathbf{M}_1] \in \mathcal{R}_q^{h \times N}$ *with* $\mathbf{M}_0 = \mathrm{diag}\big(\sigma_{k_1}(\mathbf{a}), \ldots, \sigma_{k_h}(\mathbf{a})\big)$ *for odd* $k_1, \ldots, k_h \in \mathbb{Z}_{2n}$*, the trapdoor* $\mathbf{T}$*, a target* $\mathbf{y} \in \mathcal{R}_q^{1 \times h}$*, and a width* σ*, and outputs* $\mathbf{x} \in \mathcal{R}^N$ *with* $\mathbf{M}\mathbf{x}^\top = \mathbf{y}^\top$*.*

It satisfies the following property.

- Distribution: *if* $\sigma \geq s_1(\mathbf{T}) \cdot \omega(\sqrt{\log n})$*, then the output is within negligible statistical distance from* $D_{\Lambda_{\mathbf{y}}(\mathbf{M}),\sigma}$*.*

Proof. Write $\mathbf{a} = [\bar{\mathbf{a}} \mid \mathbf{g} - \bar{\mathbf{a}}\mathbf{T}]$ as in Lemma 4. Since σ_{k_i} is a ring automorphism that fixes $\mathbf{g}$, $\sigma_{k_i}(\mathbf{a}) = [\sigma_{k_i}(\bar{\mathbf{a}}) \mid \mathbf{g} - \sigma_{k_i}(\bar{\mathbf{a}})\sigma_{k_i}(\mathbf{T})]$. Let $\mathbf{P}$ be the permutation of the columns of $\mathbf{M}$ that lists, among the columns of $\mathbf{M}_0$, first the $\bar{m}$ leading columns of every block and then the τ trailing columns of every block, and fixes the columns of $\mathbf{M}_1$. Then $\mathbf{MP} = [\mathbf{A} \mid \mathbf{M}_1]$ with

$$\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G}_h - \bar{\mathbf{A}}\mathbf{T}_h], \qquad \bar{\mathbf{A}} := \mathrm{diag}\big(\sigma_{k_i}(\bar{\mathbf{a}})\big)_{i \in [h]}, \qquad \mathbf{T}_h := \mathrm{diag}\big(\sigma_{k_i}(\mathbf{T})\big)_{i \in [h]},$$

which has the form of Lemma 4 with trapdoor $\mathbf{T}_h$. $\mathsf{SampleLeftBlock}$ outputs $\mathbf{x} := \mathbf{x}'\mathbf{P}^\top$ for $\mathbf{x}' \leftarrow\!\$\ \mathsf{SampleLeft}(\mathbf{MP}, \mathbf{T}_h, \mathbf{y}, \sigma)$, so that $\mathbf{M}\mathbf{x}^\top = \mathbf{MP}\mathbf{x}'^\top = \mathbf{y}^\top$.

Since each σ_{k_i} permutes coefficients up to sign, it is an isometry of the coefficient embedding and $s_1(\sigma_{k_i}(\mathbf{T})) = s_1(\mathbf{T})$; as $\mathbf{T}_h$ is block-diagonal, $s_1(\mathbf{T}_h) = s_1(\mathbf{T})$. By Lemma 5, $\mathbf{x}'$ is within negligible statistical distance from $D_{\Lambda_{\mathbf{y}}(\mathbf{MP}),\sigma}$. The map $\mathbf{x}' \mapsto \mathbf{x}'\mathbf{P}^\top$ permutes coordinates; it maps $\Lambda_{\mathbf{y}}(\mathbf{MP})$ bijectively onto $\Lambda_{\mathbf{y}}(\mathbf{M})$ and preserves norms; it therefore maps $D_{\Lambda_{\mathbf{y}}(\mathbf{MP}),\sigma}$ onto $D_{\Lambda_{\mathbf{y}}(\mathbf{M}),\sigma}$. □

Lattice assumptions. Let χ_s and χ be distributions of small elements of $\mathcal{R}$. Ring-LWE [LPR10] asserts that $(\mathbf{a}, \bar{s}\mathbf{a} + \mathbf{e}) \approx_c (\mathbf{a}, \mathbf{u})$ for $\bar{s} \leftarrow\!\$\ \chi_s$, $\mathbf{a}, \mathbf{u} \leftarrow\!\$\ \mathcal{R}_q^{1 \times m}$, and $\mathbf{e} \leftarrow\!\$\ \chi^{1 \times m}$. Our ciphertexts additionally contain *automorphism keys*, i.e., key-switching keys from $\sigma_k(\bar{s})$ to $\bar{s}$, as published by FHE schemes that support rotations [GHS12, HS14].

Assumption 1 (Circular Ring-LWE with automorphism keys)**.** *For* $\bar{s} \leftarrow\!\$\ \chi_s$*,* $\mathbf{a} \leftarrow\!\$\ \mathcal{R}_q^{1 \times m}$*,* $\mathbf{a}_k \leftarrow\!\$\ \mathcal{R}_q^{1 \times \tau}$*, and noise from* χ*,*

$$\Big(\mathbf{a},\ \bar{s}\mathbf{a} + \mathbf{e},\ \big\{\mathbf{a}_k,\ \bar{s}\mathbf{a}_k - \sigma_k(\bar{s})\mathbf{g} + \mathbf{e}_k\big\}_{k \in \mathcal{J}}\Big) \approx_c \Big(\mathbf{a},\ \mathbf{u},\ \big\{\mathbf{a}_k,\ \mathbf{u}_k\big\}_{k \in \mathcal{J}}\Big),$$

where $\mathbf{u} \leftarrow\!\$\ \mathcal{R}_q^{1 \times m}$ *and* $\mathbf{u}_k \leftarrow\!\$\ \mathcal{R}_q^{1 \times \tau}$*.*

The assumption is circular: the automorphism keys encrypt $\sigma_k(\bar{s})$ under $\bar{s}$ itself. It has been widely used in practical FHE schemes [BGV12, Bra12, FV12, HS14, CKKS17], which publish the same keys, often called rotation or Galois keys. When no confusion arises, we refer to it simply as *circular Ring-LWE*.

3.3 Predicate Encryption

Let $\mathcal{F}$ be a class of functions $F : \mathcal{X} \to \{0, 1\}$. A key sk_F is *authorized* for an attribute x if $F(x) = 0$, and *unauthorized* otherwise. A PE scheme for $\mathcal{F}$ with message space $\mathcal{M}$ consists of the following PPT algorithms.

- $\mathsf{Setup}(1^\lambda, \mathcal{F}) \to (\mathsf{mpk}, \mathsf{msk})$: takes the security parameter and the function class, and outputs a master public key and a master secret key. The other algorithms implicitly take λ and $\mathcal{F}$.
- $\mathsf{KeyGen}(\mathsf{msk}, F) \to \mathsf{sk}_F$: takes the master secret key and a function $F \in \mathcal{F}$, and outputs a secret key for F.
- $\mathsf{Enc}(\mathsf{mpk}, x, m) \to \mathsf{ct}$: takes the master public key, an attribute $x \in \mathcal{X}$, and a message $m \in \mathcal{M}$, and outputs a ciphertext.

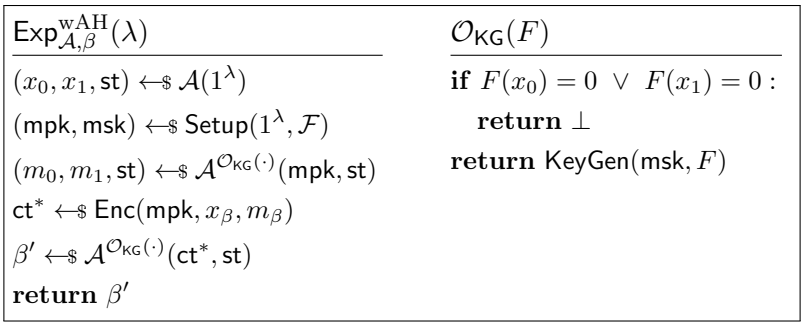

Figure 3: The selective weak attribute-hiding experiment. The adversary may query only keys that are unauthorized for both challenge attributes.

- $\mathsf{Dec}(\mathsf{sk}_F, F, \mathsf{ct}) \to m'$: takes a secret key, its function, and a ciphertext, and outputs a message $m' \in \mathcal{M}$.

The scheme is *correct* if $\Pr[\mathsf{Dec}(\mathsf{sk}_F, F, \mathsf{Enc}(\mathsf{mpk}, x, m)) = m] \geq 1 - \mathrm{negl}(\lambda)$ for all $F \in \mathcal{F}$, $x \in \mathcal{X}$ with $F(x) = 0$, and $m \in \mathcal{M}$, where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda, \mathcal{F})$ and $\mathsf{sk}_F \leftarrow\!\$\ \mathsf{KeyGen}(\mathsf{msk}, F)$.

Definition 1 (Selective weak attribute hiding)**.** For $\beta \in \{0, 1\}$ and an adversary $\mathcal{A}$, let $\mathsf{Exp}^{\mathrm{wAH}}_{\mathcal{A},\beta}(\lambda)$ be the experiment in Figure 3. The scheme is *selectively weakly attribute-hiding* if, for every PPT adversary $\mathcal{A}$,

$$\left|\Pr\left[\mathsf{Exp}^{\mathrm{wAH}}_{\mathcal{A},0}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathrm{wAH}}_{\mathcal{A},1}(\lambda) = 1\right]\right| = \mathrm{negl}(\lambda).$$

3.4 Branching Programs

We consider layered branching programs whose state transitions are given as lookup tables.

Definition 2 (Branching programs)**.** A *branching program* F of length T and width w over an alphabet Σ, which we identify with $[0, |\Sigma| - 1]$, on attributes of length ℓ consists of a read map $\iota : [0, T-1] \to [\ell]$ and transition tables

$$f_i : [0, w-1] \times \Sigma \to [0, w-1] \quad (\forall i \in [0, T-2]), \qquad f_{T-1} : [0, w-1] \times \Sigma \to \{0, 1\}.$$

On an attribute $x \in \Sigma^\ell$, it computes $z_0 := 0$ and $z_{i+1} := f_i(z_i, x_{\iota(i)})$ for all $i \in [0, T-1]$, and outputs $F(x) := z_T$. Its *lookup width* is $W := w|\Sigma|$, padded to a power of two dividing n and strictly smaller than n.

Since the read map is arbitrary, a symbol may be read many times. A key for F is authorized for x when $F(x) = 0$; accordingly, the last table maps accepting configurations to 0.

Examples. The following predicates are branching programs in the sense of Definition 2.

- *DFAs and regular expressions.* A DFA run on strings of length ℓ is a branching program with $T = \ell$, $\iota(i) = i + 1$, $f_i = \delta$ for its transition function δ, and a last table that maps accepting states to 0; regular expressions compile into DFAs.
- *Comparisons and ranges.* Comparing a hidden integer with a threshold in the key, from the most significant bit, gives a width-3 branching program; each layer uses the corresponding threshold bit. A range combines two such comparisons into a width-5 branching program, merging all states outside the range into one rejecting state.

- *Hamming-distance thresholds.* Counting mismatches with a reference string in the key, saturating at $\rho+1$, gives a branching program of width $w=\rho+2$ for the threshold ρ; each layer uses the corresponding reference bit.
- *Boolean formulas.* A circuit of depth d with fan-in two is a branching program of length at most 4^d with $|\Sigma|=2$ and $w=5$ [Bar89] or $w=8$ [BOC92], so that $W=16$.

Several bits may also be read at once as one symbol of a larger alphabet, which shortens the program at the cost of a larger W.

Heterogeneous predicates. The predicates above can be combined into a single branching program, for instance by running them one after another while keeping in the state whether the previous ones accept. One key thus enforces their conjunction, such as a range on a timestamp field together with a regular expression on an event field.

4 Private Computation over Key-Homomorphic Encodings

This section shows how to evaluate a branching program on encodings of hidden values and, in a corresponding way, on public matrices: we define primitive operations over encodings, combine them into lookup tables, and use one multiplication and one table per step of the program.

4.1 Encodings and Gadget Ciphertexts

A secret $\bar{s}\in\mathcal{R}_q$ defines $\mathbf{s}:=(\bar{s},1)\in\mathcal{R}_q^{1\times 2}$ and, for $k\in\mathcal{H}$, $\mathbf{s}_k:=\sigma_k(\mathbf{s})=(\sigma_k(\bar{s}),1)$. We use the following declarations for BGG+ encodings [BGG+14]. For $\mathbf{c}\in\mathcal{R}_q^{1\times m}$, a mask secret $\mathbf{s}'\in\mathcal{R}_q^{1\times 2}$, a public matrix $\mathbf{A}\in\mathcal{R}_q^{2\times m}$, and a payload vector $\mathbf{x}\in\mathcal{R}_q^{1\times 2}$,

$$\mathbf{c}:[\![\mathbf{x}]\!]_{\mathbf{s}'}^{\mathbf{A}} \qquad \text{declares} \qquad \mathrm{err}(\mathbf{c}):=\|\mathbf{c}-\mathbf{s}'\mathbf{A}+\mathbf{x}\mathbf{G}\|_\infty.$$

More generally, for a payload row $\mathbf{p}$, $\mathbf{c}:\langle\mathbf{p}\rangle_{\mathbf{s}'}^{\mathbf{A}}$ declares $\mathrm{err}(\mathbf{c}):=\|\mathbf{c}-\mathbf{s}'\mathbf{A}+\mathbf{p}\|_\infty$, so that $[\![\mathbf{x}]\!]$ is shorthand for $\langle\mathbf{x}\mathbf{G}\rangle$. An encoding of an attribute $\mu\in\mathcal{R}_q$ under $\mathbf{s}$ is $\mathbf{c}:[\![\mu\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}}$, i.e., $\mathbf{c}=\mathbf{s}(\mathbf{A}-\mu\mathbf{G})+\mathbf{e}$.

For $\mathbf{C}\in\mathcal{R}_q^{2\times m}$, a secret $\mathbf{u}\in\mathcal{R}_q^{1\times 2}$, and $\mathbf{f}\in\mathcal{R}_q^{1\times m}$, the declaration $\mathbf{C}:\mathsf{GC}_{\mathbf{u}}\langle\mathbf{f}\rangle$ defines $\mathrm{err}(\mathbf{C}):=\|\mathbf{u}\mathbf{C}-\mathbf{f}\|_\infty$, and $\mathbf{C}:\mathsf{GC}_{\mathbf{u}}(\mathbf{z})$ stands for $\mathbf{C}:\mathsf{GC}_{\mathbf{u}}\langle\mathbf{z}\mathbf{G}\rangle$. We call $\mathbf{C}$ a *gadget ciphertext*. For $\mathbf{z}=\nu\mathbf{u}$ it has the form of a GSW ciphertext [GSW13] of ν, but we never compute on gadget ciphertexts homomorphically: each one is consumed by multiplications of encodings.

4.2 Evaluation Interface and Primitive Operations

Fix a secret $\mathbf{s}=(\bar{s},1)$. Every operation op consists of a public-key algorithm $\mathsf{EvalPK}_{\mathsf{op}}$, which maps the input public matrices and public parameters to an output public matrix, and a ciphertext algorithm $\mathsf{EvalCT}_{\mathsf{op}}$, which maps the input public matrices, the input encodings, and public ciphertext data to an output encoding. The operation is correct if the inputs $\mathbf{c}_i:[\![\mu_i\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}_i}$ yield an output $\mathbf{c}:[\![\mu\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}}$ with $\mathbf{A}=\mathsf{EvalPK}_{\mathsf{op}}(\{\mathbf{A}_i\})$, where μ is the intended function of the μ_i and $\mathbf{A}$ does not depend on the μ_i or on the ciphertext. We describe four operations.

Addition and scalar multiplication. For encodings $\mathbf{c}_1,\mathbf{c}_2$ of μ_1,μ_2 and a public $\alpha\in\mathcal{R}_q$,

$$\mathsf{Add}(\mathbf{c}_1,\mathbf{c}_2):=\mathbf{c}_1+\mathbf{c}_2:[\![(\mu_1+\mu_2)\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}_1+\mathbf{A}_2},$$
$$\mathsf{SMul}_\alpha(\mathbf{c}_1):=\mathbf{c}_1\mathbf{G}^{-1}(\alpha\mathbf{G}):[\![\alpha\mu_1\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}_1\mathbf{G}^{-1}(\alpha\mathbf{G})},$$

with $\mathrm{err}(\mathsf{Add})\le\mathrm{err}(\mathbf{c}_1)+\mathrm{err}(\mathbf{c}_2)$ and $\mathrm{err}(\mathsf{SMul}_\alpha)\le 2\tau n\,\mathrm{err}(\mathbf{c}_1)$, independently of the size of α. The second identity holds because $\mu_1\mathbf{s}\mathbf{G}\mathbf{G}^{-1}(\alpha\mathbf{G})=\alpha\mu_1\mathbf{s}\mathbf{G}$.

Multiplication. Let $\mathbf{s}' \in \{\mathbf{s}_k\}_{k\in\mathcal{H}}$ and let $\mathbf{C} : \mathsf{GC}_{\mathbf{s}'}(\mathbf{z})$ with $\mathbf{C} \in \mathcal{R}_q^{2\times 2\tau}$. Write each entry as $C_{r,j} = \sum_{i<\tau} 2^i d_{r,j,i}$ with binary $d_{r,j,i} \in \mathcal{R}$. For public matrices $\mathbf{A}_{r,j,i}$ fixed independently of $\mathbf{C}$, the *RHS data* of $\mathbf{C}$ are

$$\mathsf{RHS}_{\mathbf{s}'}(\mathbf{C}) := \Big(\mathbf{C},\ \big\{\mathbf{c}_{r,j,i} : [\![d_{r,j,i}\mathbf{s}']\!]_{\mathbf{s}'}^{\mathbf{A}_{r,j,i}}\big\}_{r,j,i}\Big).$$

Given $\mathbf{c} : [\![\mu\mathbf{s}']\!]_{\mathbf{s}'}^{\mathbf{A}}$, define for each column j

$$\mathbf{c}'_j := \sum_{r=1}^{2}\sum_{i<\tau}\big(\mathbf{c}_{r,j,i}\mathbf{G}^{-1}(\mathbf{A}) + d_{r,j,i}\mathbf{c}\big)\mathbf{G}^{-1}(2^i\mathbf{1}_r^\top),$$

$$\mathbf{A}'_j := \sum_{r=1}^{2}\sum_{i<\tau}\mathbf{A}_{r,j,i}\mathbf{G}^{-1}(\mathbf{A})\mathbf{G}^{-1}(2^i\mathbf{1}_r^\top),$$

and $\mathsf{Mul}(\mathbf{c}, \mathsf{RHS}_{\mathbf{s}'}(\mathbf{C})) := (\mathbf{c}'_j)_j$. This works for two reasons.

- Each summand is a known-value multiplication followed by a gadget product. The payload $d_{r,j,i}\mathbf{s}'\mathbf{A}$ of $\mathbf{c}_{r,j,i}\mathbf{G}^{-1}(\mathbf{A})$ cancels the mask term of $d_{r,j,i}\mathbf{c}$, which leaves the payload $d_{r,j,i}\mu\mathbf{s}'\mathbf{G}$.
- The weights 2^i reassemble the entries of $\mathbf{C}$: $\sum_{r,i} 2^i d_{r,j,i}(\mathbf{s}')_r = (\mathbf{s}'\mathbf{C})_j$.

Hence

$$\mathsf{Mul}\big(\mathbf{c}, \mathsf{RHS}_{\mathbf{s}'}(\mathbf{C})\big) : [\![\mu\mathbf{z}]\!]_{\mathbf{s}'}^{\mathbf{A}'}, \qquad \mathrm{err} \le 2\tau n\big(2\tau\beta_{\mathsf{RHS}} + \mathrm{err}(\mathbf{c})\big) + \|\mu\|_1\,\mathrm{err}(\mathbf{C}), \tag{1}$$

where $\beta_{\mathsf{RHS}} := \max \mathrm{err}(\mathbf{c}_{r,j,i})$ and $\mathbf{A}' = (\mathbf{A}'_j)_j$ depends only on $\mathbf{A}$ and the $\mathbf{A}_{r,j,i}$. The term $\|\mu\|_1\,\mathrm{err}(\mathbf{C})$ stays small when μ is a monomial. Since $\mathbf{G}^{-1}(2^i\mathbf{1}_r^\top)$ selects a single column, $\mathbf{c}'_j$ costs 2τ inner products of length 2τ, and one Mul, with 2τ columns, therefore costs $8\tau^3$ ring multiplications.

We use Mul in two ways. With $\mathbf{s}' = \mathbf{s}$ and an attribute symbol ν supplied as $\mathbf{C} : \mathsf{GC}_{\mathbf{s}}(\nu\mathbf{s})$, it multiplies the encoded value by ν: $\mathsf{Mul}(\mathbf{c}, \mathsf{RHS}_{\mathbf{s}}(\mathbf{C})) : [\![\mu\nu\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}'}$. With $\mathbf{s}' = \mathbf{s}_k$ and the gadget ciphertext $\mathbf{C}_k$ below, it switches the payload secret in an automorphism.

Automorphisms. For $k \in \mathcal{J}$, the ciphertext provides a mask-switching helper

$$\mathbf{h}_k : [\![\mathbf{s}_k]\!]_{\mathbf{s}}^{\mathbf{H}_k}, \qquad \text{i.e.,} \qquad \mathbf{h}_k = \mathbf{s}\mathbf{H}_k - \mathbf{s}_k\mathbf{G} + \mathbf{e}_k,$$

for a public $\mathbf{H}_k \in \mathcal{R}_q^{2\times 2\tau}$. Applying σ_k to $\mathbf{h}_{k^{-1}}$ gives $\sigma_k(\mathbf{h}_{k^{-1}}) = \mathbf{s}_k\sigma_k(\mathbf{H}_{k^{-1}}) - \mathbf{s}\mathbf{G} + \sigma_k(\mathbf{e}_{k^{-1}})$. Since the second entry of $\mathbf{s}_k$ is 1, the matrix

$$\mathbf{C}_k := \sigma_k(\mathbf{H}_{k^{-1}}) - \begin{pmatrix}\mathbf{0}\\ \sigma_k(\mathbf{h}_{k^{-1}})\end{pmatrix} : \mathsf{GC}_{\mathbf{s}_k}(\mathbf{s}), \qquad \mathrm{err}(\mathbf{C}_k) = \|\mathbf{e}_{k^{-1}}\|_\infty, \tag{2}$$

is a gadget ciphertext that switches the payload secret from $\mathbf{s}_k$ to $\mathbf{s}$, computable by anyone. For $k \in \mathcal{K}$, the ciphertext provides its RHS data $\mathsf{RHS}_{\mathbf{s}_k}(\mathbf{C}_k)$ under public matrices $\mathbf{A}^{\mathrm{aut}}_{k,r,j,i}$; since $k, k^{-1} \in \mathcal{J}$, the helpers needed by Aut_k are available. For $\mathbf{c} : [\![\mu\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}}$, we have $\sigma_k(\mathbf{c}) : [\![\sigma_k(\mu)\mathbf{s}_k]\!]_{\mathbf{s}_k}^{\sigma_k(\mathbf{A})}$, and define

$$\mathbf{r}_k := \mathsf{Mul}\big(\sigma_k(\mathbf{c}), \mathsf{RHS}_{\mathbf{s}_k}(\mathbf{C}_k)\big) : [\![\sigma_k(\mu)\mathbf{s}]\!]_{\mathbf{s}_k}^{\mathbf{A}'_k},$$

$$\mathsf{Aut}_k(\mathbf{c}) := \mathbf{h}_k\mathbf{G}^{-1}(\mathbf{A}'_k) + \mathbf{r}_k : [\![\sigma_k(\mu)\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{H}_k\mathbf{G}^{-1}(\mathbf{A}'_k)}.$$

The first step switches the payload secret by (1). In the second, the payload $\mathbf{s}_k\mathbf{A}'_k$ of $\mathbf{h}_k\mathbf{G}^{-1}(\mathbf{A}'_k)$ cancels the mask term of $\mathbf{r}_k$. Thus Aut_k maps an encoding of μ to an encoding of $\sigma_k(\mu)$ under $\mathbf{s}$, with

$$\mathrm{err}(\mathsf{Aut}_k(\mathbf{c})) \le 2\tau n\ \mathrm{err}(\mathbf{h}_k) + 2\tau n\big(2\tau\beta_{\mathsf{RHS}} + \mathrm{err}(\mathbf{c})\big) + \|\mu\|_1\,\mathrm{err}(\mathbf{h}_{k^{-1}}).$$

The public matrix $\mathbf{H}_k\mathbf{G}^{-1}(\mathbf{A}'_k)$ depends only on $\mathbf{A}$, $\mathbf{H}_k$, and the $\mathbf{A}^{\mathrm{aut}}_{k,r,j,i}$. For $k \in \mathcal{K}$, Aut_k uses exactly the helpers $\mathbf{h}_k$ and $\mathbf{h}_{k^{-1}}$ and the RHS data $\mathsf{RHS}_{\mathbf{s}_k}(\mathbf{C}_k)$ under $\{\mathbf{A}^{\mathrm{aut}}_{k,r,j,i}\}$, which we call the *helper data* of σ_k. We set $\mathsf{Aut}_1 := \mathrm{id}$.

Composition. Applying the corresponding public-key algorithms of a sequence of these operations to public matrices defines an algorithm EvalPK, and applying the ciphertext algorithms in the same order to encodings held by the evaluator, together with their public matrices, defines EvalCT. By the identities above and induction on the sequence, the output of EvalCT is an encoding of the intended value under the output matrix of EvalPK, which depends only on the public matrices and not on the encoded values or the ciphertext. Its error is bounded by composing the bounds of the operations. The bounds of Mul and Aut_k contain the terms $\|\mu\|_1 \operatorname{err}(\mathbf{C})$ and $\|\mu\|_1 \operatorname{err}(\mathbf{h}_{k^{-1}})$, which depend on the encoded value μ; we apply these operations only to monomials $\mu = \pm X^p$, for which these terms are at most $\operatorname{err}(\mathbf{C})$ and $\operatorname{err}(\mathbf{h}_{k^{-1}})$. Since the public side does not depend on the encoded values, it is evaluated once per predicate; on the ciphertext side, the decompositions $\mathbf{G}^{-1}(\cdot)$ of the public matrices can be precomputed.

4.3 Lookup Tables

For $z \in \mathcal{R}_q$, let $\pi_W(z)$ keep the coefficients of z at positions divisible by W and set all others to zero.

Lemma 7 (Coefficient sieve [KLLL26])**.** *For every $z \in \mathcal{R}_q$, $\sum_{k\in\mathcal{H}} \sigma_k(z) = W \cdot \pi_W(z)$.*

Proof. By linearity, it suffices to consider $z = X^p$ with $0 \le p < n$. Set

$$S_W(p) := \sum_{j=0}^{W-1} X^{j(2n/W)p}.$$

We show by induction over powers of two W that $S_W(p) = W$ if $W \mid p$, and 0 otherwise. The case $W = 1$ is immediate. For $W > 1$, pairing the terms indexed by j and $j + W/2$ and using $X^n = -1$ gives

$$\begin{aligned} S_W(p) &= (1 + X^{np}) \sum_{j=0}^{W/2-1} X^{j(2n/W)p} \\ &= \begin{cases} 0 & \text{if } p \text{ is odd,} \\ 2S_{W/2}(p/2) & \text{if } p \text{ is even.} \end{cases} \end{aligned}$$

For odd p, we have $W \nmid p$; for even p, the claim follows by induction since $W \mid p$ if and only if $W/2 \mid p/2$. Finally, the definition of $\mathcal{H}$ gives

$$\sum_{k\in\mathcal{H}} \sigma_k(X^p) = X^p S_W(p) = W\pi_W(X^p). \qquad \square$$

Lemma 8 (Lookup tables)**.** *For a table $L : [0, W-1] \to [0, W-1]$ and $k \in \mathcal{H}$, let*

$$\alpha_{k,L} := W^{-1} \sum_{j=0}^{W-1} X^{L(j)} \sigma_k(X^{-j}), \qquad v_{k,L} := W^{-1} \sum_{j=0}^{W-1} L(j)\, \sigma_k(X^{-j}),$$

where W^{-1} is taken modulo q. Then for every $u \in [0, W-1]$,

$$\sum_{k\in\mathcal{H}} \alpha_{k,L}\, \sigma_k(X^u) = X^{L(u)}, \qquad \sum_{k\in\mathcal{H}} v_{k,L}\, \sigma_k(X^u) = L(u).$$

Moreover, let $b := \log_2 W$ and let $\mathcal{K} = \{g_0, \dots, g_{b-1}\}$ be the binary basis of $\mathcal{H}$ fixed in Section 3. Let $\mathbf{c} : [\![X^u \mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}}$, and suppose that, for every $k \in \mathcal{K}$, the evaluator holds

- *the helpers* $\mathbf{h}_{k'} : [\![\mathbf{s}_{k'}]\!]_{\mathbf{s}}^{\mathbf{H}_{k'}}$ *for* $k' \in \{k, k^{-1}\}$*, i.e.,* $\mathbf{h}_{k'} = \mathbf{s}\mathbf{H}_{k'} - \mathbf{s}_{k'}\mathbf{G} + \mathbf{e}_{k'}$*, with* $\|\mathbf{e}_{k'}\|_\infty \le B_\chi$*, and*
- *the RHS data* $\mathsf{RHS}_{\mathbf{s}_k}(\mathbf{C}_k)$ *of the gadget ciphertext* $\mathbf{C}_k$ *of* (2) *under public matrices* $\{\mathbf{A}^{\mathrm{aut}}_{k,r,j,i}\}$*, with digit encodings of error at most* β_{RHS}*.*

For $j \le b$*, let* $\mathcal{H}_j := \{\prod_{i<j} g_i^{e_i} : e_i \in \{0,1\}\}$*, so that* $\mathcal{H}_0 = \{1\}$ *and* $\mathcal{H}_b = \mathcal{H}$*. Let* $\mathbf{c}_{\sigma_1} := \mathbf{c}$*, and in rounds* $j = 0, \dots, b-1$*, let*

$$\mathbf{c}_{\sigma_{k\cdot g_j}} := \mathsf{Aut}_{g_j}(\mathbf{c}_{\sigma_k}) \qquad \text{for every } k \in \mathcal{H}_j,$$

so that $\mathbf{c}_{\sigma_k}$ *is an encoding of* $\sigma_k(X^u)$ *for every* $k \in \mathcal{H}$*. Then*

$$\mathsf{LUT}_L(\mathbf{c}) := \sum_{k\in\mathcal{H}} \mathsf{SMul}_{\alpha_{k,L}}(\mathbf{c}_{\sigma_k}) : [\![X^{L(u)}\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}_L},$$

where $\mathbf{A}_L$*, computed from* $\mathbf{A}$ *by the corresponding public-key algorithms, depends only on* $\mathbf{A}$*, the matrices* $\mathbf{H}_{k'}$ *and* $\mathbf{A}^{\mathrm{aut}}_{k,r,j,i}$*, and* L*, and* $\mathrm{err}(\mathsf{LUT}_L(\mathbf{c})) \le W(2\tau n)^{b+1}\big(\mathrm{err}(\mathbf{c}) + 4\tau\beta_{\mathsf{RHS}} + 4B_\chi\big)$*. The same holds for* $\mathsf{LUT}^{\mathrm{coef}}_L$*, defined with* $v_{k,L}$ *in place of* $\alpha_{k,L}$*, with payload* $L(u)\mathbf{s}$*.*

Proof. Selection. Let $u, j \in [0, W-1]$. Since $W \mid n$, the monomial X^{u-j} is $\pm X^p$ for a position $p \equiv u - j \pmod W$. As $|u-j| < W$, this position is divisible by W if and only if $u = j$, in which case $X^{u-j} = 1$. Hence $\pi_W(X^{u-j})$ equals 1 if $u = j$ and 0 otherwise.

The identities. Using $\sigma_k(X^{-j})\sigma_k(X^u) = \sigma_k(X^{u-j})$ and Lemma 7,

$$\sum_{k\in\mathcal{H}} \alpha_{k,L}\,\sigma_k(X^u) = W^{-1}\sum_{j=0}^{W-1} X^{L(j)} \sum_{k\in\mathcal{H}} \sigma_k(X^{u-j}) = \sum_{j=0}^{W-1} X^{L(j)}\,\pi_W(X^{u-j}) = X^{L(u)},$$

where W is invertible modulo q because q is odd. The same computation with $L(j)$ in place of $X^{L(j)}$ gives the second identity.

Rounds. By induction on j, after round j every $\mathbf{c}_{\sigma_k}$ with $k \in \mathcal{H}_{j+1}$ is an encoding of $\sigma_k(X^u)$, since $\mathcal{H}_{j+1} = \mathcal{H}_j \cup \mathcal{H}_j \cdot g_j$ and $\sigma_{g_j} \circ \sigma_k = \sigma_{k\cdot g_j}$. As the basis is binary, $\mathcal{H}_b = \mathcal{H}$, and every $\mathbf{c}_{\sigma_k}$ is defined exactly once. By the scalar-multiplication and addition rules of Subsection 4.2, the weighted sum of these encodings encodes $\sum_{k\in\mathcal{H}} \alpha_{k,L}\sigma_k(X^u) = X^{L(u)}$, as required. Its public matrix $\mathbf{A}_L$ is computed by composing the corresponding public-key algorithms.

Error. Every intermediate value $\sigma_k(X^u)$ is a signed monomial $\pm X^p$ with $0 \le p < n$, so its coefficient ℓ_1-norm is 1. The bound of Aut_k therefore maps an error E to at most $2\tau n(E + 2\tau\beta_{\mathsf{RHS}} + 2B_\chi)$. Each $\mathbf{c}_{\sigma_k}$ is obtained from $\mathbf{c}$ by at most b automorphisms; since $2\tau n \ge 2$, its error is at most $(2\tau n)^b E + (2\tau\beta_{\mathsf{RHS}} + 2B_\chi)\sum_{t=1}^{b}(2\tau n)^t \le (2\tau n)^b\big(E + 4\tau\beta_{\mathsf{RHS}} + 4B_\chi\big)$ for $E = \mathrm{err}(\mathbf{c})$. We then multiply each of the W encodings by its public coefficient using SMul, increasing its error bound by a factor of at most $2\tau n$. □

Since $\sigma_k(X^u) = (X^u)^k$, the lemma expresses every table as a polynomial in X^u with public coefficients. The coefficients $\alpha_{k,L}$ depend only on the table and are applied by the evaluator as public scalar multiplications $\mathsf{SMul}_{\alpha_{k,L}}$, both in EvalPK and in EvalCT. The gadget ciphertexts supplied by the encryptor, namely $\mathbf{C}_k$, which only switches the secret back from $\mathbf{s}_k$ to $\mathbf{s}$, and the $\mathbf{C}^{\mathrm{in}}_t$ of the attribute symbols, do not involve L. In particular, the ciphertext does not depend on the predicate that is later evaluated on it.

4.4 Branching Programs over Encodings

We now combine the operations to evaluate a branching program F (Definition 2) over encodings. Recall that F has w states $[0, w-1]$, reads symbols from the alphabet $\Sigma = [0, |\Sigma|-1]$, and has lookup width $W = w|\Sigma|$, padded to a power of two dividing n and strictly smaller than n, which is the size of the tables below. Step i reads the symbol $x_{\iota(i)}$, i.e., $\iota(i) \in [\ell]$ is the index of the attribute symbol read at step i. Encode each symbol as $\nu_t := X^{wx_t}$ for $t \in [\ell]$, and define the tables $L_i : [0, W-1] \to [0, W-1]$ by $L_i(z + wx) := f_i(z, x)$ for $(z, x) \in [0, w-1] \times \Sigma$ and arbitrarily elsewhere. The evaluations take public matrices pp, which consist of a matrix $\mathbf{A}_0$, the matrices $\mathbf{A}^{\mathrm{in}}_{t,r,j,i}$ and $\mathbf{A}^{\mathrm{aut}}_{k,r,j,i}$ of digit encodings, and the matrices $\mathbf{H}_k$ of helpers, and ciphertext data ct, which consist of an encoding $\mathbf{c}_0$ of 1 under $\mathbf{A}_0$, the RHS data $\mathsf{RHS}_{\mathbf{s}}(\mathbf{C}^{\mathrm{in}}_t)$ of gadget ciphertexts of the ν_t under $\{\mathbf{A}^{\mathrm{in}}_{t,r,j,i}\}$ for $t \in [\ell]$, and, for every $k \in \mathcal{K}$, the helpers $\mathbf{h}_k, \mathbf{h}_{k^{-1}}$ and the RHS data $\mathsf{RHS}_{\mathbf{s}_k}(\mathbf{C}_k)$ under $\{\mathbf{A}^{\mathrm{aut}}_{k,r,j,i}\}$. The evaluation $\mathsf{EvalCT}(F, \mathsf{pp}, \mathsf{ct})$ starts from $\mathbf{c} := \mathbf{c}_0$ and computes, for $i = 0, \ldots, T-1$,

$$\mathbf{c} \leftarrow \mathsf{Mul}\big(\mathbf{c}, \mathsf{RHS}_{\mathbf{s}}(\mathbf{C}^{\mathrm{in}}_{\iota(i)})\big), \qquad \mathbf{c} \leftarrow \mathsf{LUT}_{L_i}(\mathbf{c}),$$

where $\mathsf{LUT}_{L_{T-1}}$ is replaced by $\mathsf{LUT}^{\mathrm{coef}}_{L_{T-1}}$, and outputs the final $\mathbf{c}$; it uses pp to keep track of the public matrix of $\mathbf{c}$, whose decomposition $\mathbf{G}^{-1}(\cdot)$ each operation requires. The evaluation $\mathsf{EvalPK}(F, \mathsf{pp})$ applies the corresponding public-key algorithms to pp and outputs the public matrix of the final $\mathbf{c}$.

Lemma 9 (Evaluation of branching programs)**.** *Let F be a branching program of length T, width w, and lookup width W, let $b := \log_2 W$, let $x \in \Sigma^\ell$, and let $\nu_t := X^{wx_t}$ for $t \in [\ell]$. Suppose that the evaluator holds*

- *an encoding $\mathbf{c}_0 : [\![\mathbf{s}]\!]^{\mathbf{A}_0}_{\mathbf{s}}$ of 1 with error at most B_0,*
- *for every $t \in [\ell]$, a gadget ciphertext $\mathbf{C}^{\mathrm{in}}_t : \mathsf{GC}_{\mathbf{s}}(\nu_t \mathbf{s})$ with error at most $B_{\mathbf{C}}$ and its RHS data $\mathsf{RHS}_{\mathbf{s}}(\mathbf{C}^{\mathrm{in}}_t)$ under public matrices $\{\mathbf{A}^{\mathrm{in}}_{t,r,j,i}\}$, and*
- *for every $k \in \mathcal{K}$, the helpers $\mathbf{h}_k$ and $\mathbf{h}_{k^{-1}}$ with errors at most B_χ and the RHS data $\mathsf{RHS}_{\mathbf{s}_k}(\mathbf{C}_k)$ under $\{\mathbf{A}^{\mathrm{aut}}_{k,r,j,i}\}$, as in Lemma 8,*

where all digit encodings have errors at most β_{RHS}; let ct consist of these data, and let pp consist of $\mathbf{A}_0$, the $\mathbf{A}^{\mathrm{in}}_{t,r,j,i}$, the $\mathbf{H}_k$, and the $\mathbf{A}^{\mathrm{aut}}_{k,r,j,i}$. Then

$$\mathbf{c}_F := \mathsf{EvalCT}(F, \mathsf{pp}, \mathsf{ct}) : [\![F(x)\,\mathbf{s}]\!]^{\mathbf{A}_F}_{\mathbf{s}}, \qquad \mathbf{A}_F := \mathsf{EvalPK}(F, \mathsf{pp}),$$

where $\mathbf{A}_F$ depends only on F and the public matrices, and

$$\mathrm{err}(\mathbf{c}_F) \le \gamma^T\big(B_0 + \Delta\big), \qquad \gamma := W(2\tau n)^{b+2}, \quad \Delta := W(2\tau n)^{b+1}\big(4\tau^2 n\beta_{\mathsf{RHS}} + B_{\mathbf{C}} + 4\tau\beta_{\mathsf{RHS}} + 4B_\chi\big).$$

Proof. Correctness. We show by induction on i that before step i the encoding $\mathbf{c}$ encodes X^{z_i}. For $i = 0$, this holds since $z_0 = 0$. For step i, let $u_i := z_i + wx_{\iota(i)} \in [0, W-1]$.

- The multiplication gives $X^{z_i} \cdot X^{wx_{\iota(i)}} = X^{u_i}$.
- If $i < T-1$, Lemma 8 gives an encoding of $X^{L_i(u_i)} = X^{f_i(z_i, x_{\iota(i)})} = X^{z_{i+1}}$.
- If $i = T-1$, $\mathsf{LUT}^{\mathrm{coef}}_{L_{T-1}}$ gives an encoding of $f_{T-1}(z_{T-1}, x_{\iota(T-1)}) = z_T = F(x)$.

Since every multiplication and automorphism is applied to an encoding of a monomial, the composition argument of Subsection 4.2 gives the declaration of $\mathbf{c}_F$ with a public matrix $\mathbf{A}_F$ independent of x.

Error. Let E_i bound the error before step i, with $E_0 \le B_0$.

- By (1), the multiplication yields error at most $E' \le 2\tau n E_i + 4\tau^2 n\beta_{\mathsf{RHS}} + B_{\mathbf{C}}$.
- By Lemma 8, the table yields error at most $W(2\tau n)^{b+1}(E' + 4\tau\beta_{\mathsf{RHS}} + 4B_\chi)$.

Together, $E_{i+1} \le W(2\tau n)^{b+2} E_i + \Delta$. Unrolling with $\gamma \ge 2$ gives $E_T \le \gamma^T E_0 + \Delta(\gamma^T - 1)/(\gamma - 1) \le \gamma^T(B_0 + \Delta)$. □

Hence $\log_2 q$ grows linearly in T, by $\log_2 \gamma$ per step for the evaluation alone. These are worst-case bounds; Section 6 selects parameters from heuristic average-case bounds.

5 Predicate Encryption for Branching Programs

5.1 Construction

The scheme supports branching programs (Definition 2) of length T, width w, alphabet Σ, and attribute length ℓ, with lookup width $W < n$. We use the group $\mathcal{H}$, its binary basis $\mathcal{K}$, and $\mathcal{J} = \mathcal{K} \cup \mathcal{K}^{-1}$ of Section 3, with $b = \log_2 W$, and a secret distribution χ_s of small elements of $\mathcal{R}$. Recall that $\mathsf{EvalPK}(F, \mathsf{pp})$ and $\mathsf{EvalCT}(F, \mathsf{pp}, \mathsf{ct})$ are the evaluations of F of Subsection 4.4; below, they are applied to the corresponding components of mpk and of a ciphertext.

Parameters. Let $q = \prod_{i=1}^{L_q} q_i$ for distinct primes $q_i \equiv 1 \pmod{2n}$, and let $m_B = \bar{m} + \tau$. Use $\chi := D_{\mathcal{R},\sigma_\chi}$ for base noise, $\chi_{\mathrm{fl}} := D_{\mathcal{R},\sigma_{\mathrm{fl}}}$ for fresh-encoding errors, and Gaussian widths $\sigma_{\mathbf{T}}$ and σ for the trapdoor and keys. Choose these parameters to satisfy circular Ring-LWE hardness and the uniformity, flooding, sampling, and correctness conditions stated in Appendix A. The modulus bit size can be chosen as

$$\log_2 q = O\Big(\lambda + (T+1)\log_2\big(W(2\tau n)^{b+2}\big)\Big).$$

Section 6 gives the concrete parameter choices and implementation relaxations.

Setup. $\mathsf{Setup}(1^\lambda, \mathcal{F})$ samples the following matrices.

- *Public matrix of the anchor:* $(\mathbf{B}^{(1)}, \mathbf{T}) \leftarrow\!\$\ \mathsf{TrapGen}(1^n, q, 1, \bar{m}, \sigma_{\mathbf{T}})$ with $\mathbf{B}^{(1)} \in \mathcal{R}_q^{1\times m_B}$, and $\mathbf{B}^{(2)} \leftarrow\!\$\ \mathcal{R}_q^{1\times m_B}$; let $\mathbf{B} := \binom{\mathbf{B}^{(1)}}{\mathbf{B}^{(2)}}$.
- *Public matrices of helpers:* $\mathbf{H}_k \leftarrow\!\$\ \mathcal{R}_q^{2\times 2\tau}$ for every $k \in \mathcal{J}$.
- *Ciphertext rows:* $\mathbf{a}_t \leftarrow\!\$\ \mathcal{R}_q^{1\times 2\tau}$ for every $t \in [\ell]$, the first rows of the attribute ciphertexts.
- *Public matrices of encodings:* $\mathbf{A}_0 \leftarrow\!\$\ \mathcal{R}_q^{2\times 2\tau}$ for the initial state, and the digit matrices $\mathbf{A}^{\mathrm{in}}_{t,r,j,i} \leftarrow\!\$\ \mathcal{R}_q^{2\times 2\tau}$ for every $t \in [\ell]$ and $\mathbf{A}^{\mathrm{aut}}_{k,r,j,i} \leftarrow\!\$\ \mathcal{R}_q^{2\times 2\tau}$ for every $k \in \mathcal{K}$, where $(r, j, i) \in [2] \times [2\tau] \times [0, \tau - 1]$.
- *Public vector of the message:* $\mathbf{d} = (d_1, d_2) \leftarrow\!\$\ \mathcal{R}_q^{1\times 2}$, which masks the message in c_{m}.

It outputs $\mathsf{msk} := \mathbf{T}$ and $\mathsf{mpk} := (\mathbf{B}, \{\mathbf{H}_k\}, \{\mathbf{a}_t\}, \mathbf{A}_0, \{\mathbf{A}^{\mathrm{in}}_{t,r,j,i}\}, \{\mathbf{A}^{\mathrm{aut}}_{k,r,j,i}\}, \mathbf{d})$.

Encryption. $\mathsf{Enc}(\mathsf{mpk}, x, m)$ takes $x \in \Sigma^\ell$ and $m \in \mathcal{R}$ with binary coefficients, samples $\bar{s} \leftarrow\!\$\ \chi_s$, sets $\mathbf{s} := (\bar{s}, 1)$, and computes the following.

1. *Anchor and helpers:* sample $\mathbf{e}_b \leftarrow\!\$\ \chi^{1\times m_B}$ and $\mathbf{e}_k \leftarrow\!\$\ \chi^{1\times 2\tau}$ for every $k \in \mathcal{J}$, and compute

$$\mathbf{b} := \mathbf{s}\mathbf{B} + \mathbf{e}_b, \qquad \mathbf{h}_k := \mathbf{s}\mathbf{H}_k - \mathbf{s}_k\mathbf{G} + \mathbf{e}_k.$$

2. *Automorphism data:* for every $k \in \mathcal{K}$, compute the gadget ciphertext

$$\mathbf{C}_k := \sigma_k(\mathbf{H}_{k^{-1}}) - \begin{pmatrix} \mathbf{0} \\ \sigma_k(\mathbf{h}_{k^{-1}}) \end{pmatrix} : \mathsf{GC}_{\mathbf{s}_k}(\mathbf{s})$$

of (2), write its entries as $(\mathbf{C}_k)_{r,j} = \sum_{i<\tau} 2^i d^{\mathrm{aut}}_{k,r,j,i}$ with binary $d^{\mathrm{aut}}_{k,r,j,i} \in \mathcal{R}$, sample $\mathbf{e}^{\mathrm{aut}}_{k,r,j,i} \leftarrow\!\$\ \chi_{\mathrm{fl}}^{1\times 2\tau}$, and compute the digit encodings

$$\mathbf{c}^{\mathrm{aut}}_{k,r,j,i} := \mathbf{s}_k\big(\mathbf{A}^{\mathrm{aut}}_{k,r,j,i} - d^{\mathrm{aut}}_{k,r,j,i}\mathbf{G}\big) + \mathbf{e}^{\mathrm{aut}}_{k,r,j,i} : [\![d^{\mathrm{aut}}_{k,r,j,i}\mathbf{s}_k]\!]^{\mathbf{A}^{\mathrm{aut}}_{k,r,j,i}}_{\mathbf{s}_k}$$

for every $(r, j, i) \in [2] \times [2\tau] \times [0, \tau - 1]$, so that $\mathsf{RHS}_{\mathbf{s}_k}(\mathbf{C}_k) = \big(\mathbf{C}_k, \{\mathbf{c}^{\mathrm{aut}}_{k,r,j,i}\}_{r,j,i}\big)$.

3. *Attribute symbols:* for every $t \in [\ell]$, sample $\mathbf{e}_t \leftarrow\!\$\ \chi_{\mathrm{fl}}^{1\times 2\tau}$, and compute, with the row $\mathbf{a}_t$ of mpk,

$$\mathbf{C}_t^{\mathrm{in}} := \begin{pmatrix} \mathbf{a}_t \\ -\bar{s}\mathbf{a}_t + X^{wx_t}\mathbf{s}\mathbf{G} + \mathbf{e}_t \end{pmatrix} : \mathsf{GC}_{\mathbf{s}}(X^{wx_t}\mathbf{s}),$$

which satisfies $\mathbf{s}\mathbf{C}_t^{\mathrm{in}} = X^{wx_t}\mathbf{s}\mathbf{G} + \mathbf{e}_t$. Write its entries as $(\mathbf{C}_t^{\mathrm{in}})_{r,j} = \sum_{i<\tau} 2^i d_{t,r,j,i}^{\mathrm{in}}$ with binary $d_{t,r,j,i}^{\mathrm{in}} \in \mathcal{R}$, sample $\mathbf{e}_{t,r,j,i}^{\mathrm{in}} \leftarrow\!\$\ \chi_{\mathrm{fl}}^{1\times 2\tau}$, and compute the digit encodings

$$\mathbf{c}_{t,r,j,i}^{\mathrm{in}} := \mathbf{s}\big(\mathbf{A}_{t,r,j,i}^{\mathrm{in}} - d_{t,r,j,i}^{\mathrm{in}}\mathbf{G}\big) + \mathbf{e}_{t,r,j,i}^{\mathrm{in}} : [\![d_{t,r,j,i}^{\mathrm{in}}\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}_{t,r,j,i}^{\mathrm{in}}}$$

for every (r, j, i), so that $\mathsf{RHS}_{\mathbf{s}}(\mathbf{C}_t^{\mathrm{in}}) = \big(\mathbf{C}_t^{\mathrm{in}}, \{\mathbf{c}_{t,r,j,i}^{\mathrm{in}}\}_{r,j,i}\big)$.

4. *Initial state and message:* sample $\mathbf{e}_0 \leftarrow\!\$\ \chi_{\mathrm{fl}}^{1\times 2\tau}$ and $e_{\mathsf{m}} \leftarrow\!\$\ \chi$, and compute

$$\mathbf{c}_0 := \mathbf{s}(\mathbf{A}_0 - \mathbf{G}) + \mathbf{e}_0 : [\![\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}_0}, \qquad c_{\mathsf{m}} := \mathbf{s}\mathbf{d}^\top + e_{\mathsf{m}} + \lfloor q/2 \rceil m.$$

The ciphertext is $\mathsf{ct} := \big(\mathbf{b}, \{\mathbf{h}_k\}, \{\mathsf{RHS}_{\mathbf{s}_k}(\mathbf{C}_k)\}, \{\mathsf{RHS}_{\mathbf{s}}(\mathbf{C}_t^{\mathrm{in}})\}, \mathbf{c}_0, c_{\mathsf{m}}\big)$, where the matrices $\mathbf{C}_k$ are omitted because the evaluator recomputes them from the helpers, and the first rows $\mathbf{a}_t$ of the matrices $\mathbf{C}_t^{\mathrm{in}}$ are omitted because they belong to mpk.

Key generation. $\mathsf{KeyGen}(\mathsf{msk}, F)$ proceeds as follows.

1. Compute the output matrix $\mathbf{A}_F := \mathsf{EvalPK}(F, \mathsf{mpk})$.
2. Form the matrices

$$\mathbf{L} := \Big[\ \big(\boldsymbol{\delta}_i^\top \otimes \sigma_i(\mathbf{B}^{(1)})\big)_{i\in\mathcal{H}} \ \Big|\ \big(\boldsymbol{\delta}_i^\top \otimes \sigma_i(\mathbf{H}_k^{(1)}) - \boldsymbol{\delta}_{i\cdot k}^\top \otimes \mathbf{G}^{(1)}\big)_{i\in\mathcal{H},\,k\in\mathcal{J}}\ \Big] \in \mathcal{R}_q^{W\times N_L}, \quad (3)$$

$$\mathbf{M}_F := \big[\ \mathbf{L}\ \big|\ \boldsymbol{\delta}_1^\top \otimes \mathbf{A}_F^{(1)}\ \big] \in \mathcal{R}_q^{W\times N},$$

where $N_L := Wm_B + 2\tau W|\mathcal{J}|$ and $N := N_L + 2\tau$.

3. Sample $\mathbf{k}_F \leftarrow \mathsf{SampleLeftBlock}(\mathbf{M}_F, \mathbf{T}, \mathbf{y}, \sigma)$ (Lemma 6) for the target $\mathbf{y} := d_1\boldsymbol{\delta}_1 \in \mathcal{R}_q^{1\times W}$, where the blocks $\boldsymbol{\delta}_i^\top \otimes \sigma_i(\mathbf{B}^{(1)})$, $i \in \mathcal{H}$, of $\mathbf{L}$ form the block-diagonal part $\mathbf{M}_0$ with $\mathbf{a} = \mathbf{B}^{(1)}$.

It outputs $\mathsf{sk}_F := \mathbf{k}_F$.

Decryption. $\mathsf{Dec}(\mathsf{sk}_F, F, \mathsf{ct})$ proceeds as follows.

1. Evaluate $\mathbf{c}_F := \mathsf{EvalCT}(F, \mathsf{mpk}, \mathsf{ct})$.
2. Form the row $\mathbf{y}_F$ and its public constant part $\boldsymbol{\kappa}_F$:

$$\mathbf{y}_F := \Big(\big(\sigma_i(\mathbf{b})\big)_{i\in\mathcal{H}},\ \big(\sigma_i(\mathbf{h}_k)\big)_{i\in\mathcal{H},k\in\mathcal{J}},\ \mathbf{c}_F\Big),$$

$$\boldsymbol{\kappa}_F := \Big(\big(\sigma_i(\mathbf{B}^{(2)})\big)_{i\in\mathcal{H}},\ \big(\sigma_i(\mathbf{H}_k^{(2)}) - \mathbf{G}^{(2)}\big)_{i\in\mathcal{H},k\in\mathcal{J}},\ \mathbf{A}_F^{(2)}\Big).$$

3. Compute $\mu' := c_{\mathsf{m}} - d_2 - (\mathbf{y}_F - \boldsymbol{\kappa}_F)\mathbf{k}_F^\top$ and output the coefficient-wise rounding of $\mu'/\lfloor q/2 \rceil$, reduced modulo 2.

Efficiency. Table 4 lists the sizes of the scheme.

- The public key is dominated by the digit matrices, $16\tau^3$ ring elements per gadget ciphertext, of which there are $\ell + b$; they can be expanded from a public seed (Subsection 6.1).
- The ciphertext is dominated by the RHS data, $8\tau^3$ ring elements per gadget ciphertext, of which there are $\ell + b$.
- The key consists of $N = Wm_B + 2\tau W|\mathcal{J}| + 2\tau \le W(m_B + 4\tau b) + 2\tau$ short ring elements.
- Decryption performs, per step, one multiplication and $W - 1$ automorphisms in b rounds, i.e., about $8W\tau^3$ ring multiplications, followed by one inner product of length N.

Table 4: Sizes in ring elements of the public key, the ciphertext, and a secret key, for ℓ attribute symbols and LUT width $W = 2^b$.

Component	Ring elements
$\mathbf{B}$, $\mathbf{H}_k$, $\mathbf{a}_t$, $\mathbf{A}_0$, $\mathbf{d}$	$2m_B + 4\tau\lvert\mathcal{J}\rvert + 2\tau\ell + 4\tau + 2$
digit matrices $\mathbf{A}^{\text{in}}_{t,r,j,i}$, $\mathbf{A}^{\text{aut}}_{k,r,j,i}$	$16\tau^3(\ell + b)$
$\mathbf{b}$	m_B
helpers $\mathbf{h}_k$	$2\tau\lvert\mathcal{J}\rvert \le 4\tau b$
RHS data of $\mathbf{C}_k$	$8\tau^3 b$
$\mathbf{C}^{\text{in}}_t$ and RHS data	$\ell(2\tau + 8\tau^3)$
$\mathbf{c}_0$, c_{m}	$2\tau + 1$
secret key $\mathbf{k}_F$	$Wm_B + 2\tau W\lvert\mathcal{J}\rvert + 2\tau$

5.2 Correctness

Let $B_\chi := \sigma_\chi\sqrt{\lambda}$ be the coefficient bound for base-noise samples from Lemma 1. Let B_F be the evaluated-encoding error bound of Lemma 9, with initial-state, digit-encoding, and input gadget-ciphertext errors all bounded by $\sigma_{\text{fl}}\sqrt{\lambda}$ and helper errors bounded by B_χ. Thus $B_F := \gamma^T(\sigma_{\text{fl}}\sqrt{\lambda}+\Delta)$, where γ and Δ are as defined in that lemma under these bounds. The correctness condition is

$$B_\chi + Nn\sigma\sqrt{\lambda}\cdot\max(B_\chi, B_F) < \lfloor q/4\rceil,$$

where N is the number of ring elements in $\mathbf{k}_F$.

Linearity in the secret. Let $\bar{\mathbf{s}}_{\mathcal{H}} := (\sigma_i(\bar{s}))_{i\in\mathcal{H}} \in \mathcal{R}_q^{1\times W}$. Each entry of $\mathbf{y}_F$ is linear in the conjugates of $\bar{s}$: for instance, $\sigma_i(\mathbf{h}_k) = \mathbf{s}_i\sigma_i(\mathbf{H}_k) - \mathbf{s}_{i\cdot k}\mathbf{G} + \sigma_i(\mathbf{e}_k)$, in which $\sigma_i(\bar{s})$ multiplies $\sigma_i(\mathbf{H}^{(1)}_k)$ and $\sigma_{i\cdot k}(\bar{s})$ multiplies $-\mathbf{G}^{(1)}$. Collecting the coefficients of each $\sigma_i(\bar{s})$, and subtracting the parts that do not depend on $\bar{s}$, gives

$$\mathbf{y}_F - \boldsymbol{\kappa}_F = \bar{\mathbf{s}}_{\mathcal{H}}\,\mathbf{M}_F - F(x)\big(\mathbf{0},\ \bar{s}\mathbf{G}^{(1)} + \mathbf{G}^{(2)}\big) + \mathbf{e}_F, \tag{4}$$

where the second term affects only the last 2τ entries and $\mathbf{e}_F$ collects the errors of the entries of $\mathbf{y}_F$, including that of $\mathbf{c}_F$.

Theorem 1 (Correctness)**.** *The scheme of Subsection 5.1 is correct in the sense of Subsection 3.3.*

Proof. Fix $F \in \mathcal{F}$, an attribute x with $F(x) = 0$, and a message m, and let $\mathsf{ct} \leftarrow\!\$\ \mathsf{Enc}(\mathsf{mpk}, x, m)$ and $\mathbf{k}_F \leftarrow\!\$\ \mathsf{KeyGen}(\mathsf{msk}, F)$. We first bound the error row $\mathbf{e}_F$ of (4). By Lemma 9, the evaluated encoding $\mathbf{c}_F : [\![F(x)\mathbf{s}]\!]^{\mathbf{A}_F}_{\mathbf{s}}$ has error at most B_F. The other entries of $\mathbf{y}_F$ are $\sigma_i(\mathbf{b})$ and $\sigma_i(\mathbf{h}_k)$, whose errors $\sigma_i(\mathbf{e}_b)$ and $\sigma_i(\mathbf{e}_k)$ have norm at most B_χ because automorphisms preserve norms. Hence $\|\mathbf{e}_F\|_\infty \le \max(B_\chi, B_F)$.

We next show that the secret cancels. Since $F(x) = 0$, the second term of (4) vanishes, and since $\mathbf{M}_F\mathbf{k}_F^\top = \mathbf{y}^\top = \boldsymbol{\delta}_1^\top \otimes d_1$ and $\bar{\mathbf{s}}_{\mathcal{H}}\boldsymbol{\delta}_1^\top = \bar{s}$, we obtain $(\mathbf{y}_F - \boldsymbol{\kappa}_F)\mathbf{k}_F^\top = \bar{s}d_1 + \mathbf{e}_F\mathbf{k}_F^\top$. By construction, $c_{\mathsf{m}} - d_2 = \bar{s}d_1 + e_{\mathsf{m}} + \lfloor q/2\rceil m$, and therefore

$$\mu' = \lfloor q/2\rceil m + e_{\mathsf{m}} - \mathbf{e}_F\mathbf{k}_F^\top.$$

Finally, we bound the remaining error. Since the key is distributed within negligible distance from $D_{\Lambda_{\mathbf{y}}(\mathbf{M}_F),\sigma}$ and the key width exceeds the smoothing parameter of $\Lambda^\perp(\mathbf{M}_F)$, Lemma 1 gives

$\|\mathbf{k}_F\|_\infty \le \sigma\sqrt{\lambda}$ except with negligible probability. Since $\mathbf{e}_F\mathbf{k}_F^\top$ is a sum of N products of ring elements, each of norm at most $n\|\mathbf{e}_F\|_\infty\|\mathbf{k}_F\|_\infty$, the correctness condition on the parameters gives

$$\|e_\mathsf{m} - \mathbf{e}_F\mathbf{k}_F^\top\|_\infty \le B_\chi + Nn\sigma\sqrt{\lambda}\cdot\max(B_\chi, B_F) < \lfloor q/4\rceil.$$

Every coefficient of μ' thus differs from the corresponding coefficient of $\lfloor q/2\rceil m \in \{0, \lfloor q/2\rceil\}$ by less than $\lfloor q/4\rceil$, and the rounding of $\mu'/\lfloor q/2\rceil$, reduced modulo 2, returns m with overwhelming probability. □

5.3 Security

Theorem 2 (Security)**.** *Under circular Ring-LWE with automorphism keys (Assumption 1), the scheme of Subsection 5.1 is selectively weakly attribute-hiding (Definition 1).*

Proof sketch. We show that, for every $\beta \in \{0,1\}$, the view of the adversary in $\mathsf{Exp}^{\mathrm{wAH}}_{\mathcal{A},\beta}$ is indistinguishable from a view in which the challenge ciphertext is independent of (x_β, m_β). Let $x^* := x_\beta$; all queried keys satisfy $F(x^*) = 1$. The argument proceeds through four hybrids.

1. *Programming.* Since x^* is chosen before Setup, the reduction samples the ciphertext secrets first, simulates the attribute ciphertexts from $(\mathbf{B}, \mathbf{b})$, and programs the public matrices with the challenge attribute and the digits of the challenge gadget ciphertexts. Every public matrix produced by EvalPK is then expressed through the matrix $\mathbf{L}$ of (3) and a short matrix known to the reduction (the simulation form of Appendix A). This change is statistical, by noise flooding and the leftover hash lemma.
2. *Keys without trapdoor.* For $F(x^*) = 1$, the simulation form of the output matrix $\mathbf{A}_F$ contains the gadget $\mathbf{G}^{(1)}$ in the row of $\bar{s}$, which together with the helper columns of $\mathbf{L}$ yields a gadget trapdoor for $\mathbf{M}_F$. Keys are sampled by $\mathsf{SamplePre}$ with this gadget trapdoor, which produces the same distribution as KeyGen without the trapdoor of $\mathbf{B}^{(1)}$.
3. *Assumption.* Only $\mathbf{b}$, the helpers, and c_m depend on $\bar{s}$ other than through each other; all other components are computed from them. They are replaced by uniform values under Assumption 1.
4. *Statistical removal.* Once these values are uniform, the leftover hash lemma makes the programmed matrices, the encodings, and the gadget ciphertexts uniform and independent of (x^*, m_β).

The final hybrid does not depend on β. The full proof is given in Appendix A. □

Remark 1 (Authorized keys)*.* Security against one additional authorized key can likely be obtained with the two-stage sampling technique of Lai, Liu, and Wang [LLW21], which [WWLW25] combines with the dual-use technique. For an authorized key, the evaluated output matrix has the form required by this technique, and decryption leakage can be hidden by a flooding term in c_m; we leave the details to future work.

6 Implementation

We implemented the evaluation of Section 4 and the scheme of Section 5 in the mxx library [Mac26], which executes lattice-based protocols written in a Rust DSL on the CPU or the GPU. The implementation is open source and available at `https://github.com/MachinaIO/mxx/tree/main`: the PE scheme is mainly implemented in `crates/pe`, and the lookup tables over exponents in `crates/exponent-lut`. This section describes the optimizations that dominate performance, and then how we instantiate the parameters, including the parameter conditions in Appendix A that we relax.

6.1 Optimizations

Public matrices from a seed. The implementation stores a fixed-size public seed and the first row $\mathbf{B}^{(1)}$ of $\mathbf{B}$ in mpk; all remaining public-key matrix entries are derived from the seed using a hash function rather than stored explicitly. This reduces public-key storage; the sizes reported in Table 1 count the expanded matrices.

RNS arithmetic and gadget. The modulus q is a product of word-size NTT-friendly primes, and ring elements are stored in the residue number system (RNS) and in NTT form. As in [CGM+18], the gadget matrix is replaced by an RNS gadget, which first splits a ring element into its residues and then decomposes each residue in a base β_{g}; this changes only the value of τ and the bound $2\tau n$ in Section 4, which becomes $\tau n\beta_{\mathrm{g}}$ with balanced base-β_{g} digits. The digits of a gadget ciphertext are the digits of this decomposition. An automorphism is a coefficient permutation with signs, which in NTT form is a permutation of evaluation points.

Radix of the lookup tables. Instead of the binary basis of Section 3, the implementation takes a radix B, a power of two with $W = B^r$, which we choose larger than 2. Using $\mathcal{H} = \langle g \rangle$, it uses the automorphisms $\sigma_{g^{a \cdot B^j}}$ for $1 \le a < B$ and $j < r$, which compute all σ_k, $k \in \mathcal{H}$, in $r = \log_B W$ rounds from $(B-1)r$ gadget ciphertexts and at most twice as many helpers. The analysis of Sections 4 and 5 and Appendix A carries over with b replaced by r and $\mathcal{K}$ by this set. The radix $B = 2$ is the binary basis, and $B = W$ applies every automorphism directly, with depth one. A larger B lowers the noise growth of a table from $W(2\tau n)^{b+1}$ toward $W(2\tau n)^2$, and hence $\log q$, at the cost of a larger ciphertext and key.

Public rows of the gadget ciphertexts. Every gadget ciphertext of the scheme has a first row that is fixed at setup and public: the first row of $\mathbf{C}_k = \sigma_k(\mathbf{H}_{k^{-1}}) - \binom{\mathbf{0}}{\sigma_k(\mathbf{h}_{k^{-1}})}$ is $\sigma_k(\mathbf{H}^{(1)}_{k^{-1}})$, and the first row of $\mathbf{C}^{\mathrm{in}}_t$ is the row $\mathbf{a}_t$ of mpk. Since these rows are fixed before key generation, the public matrix $\mathbf{A}_F$ may depend on them, and their contribution to Mul needs no digit encodings. We describe this for $\mathbf{C}_k$; it applies to $\mathbf{C}^{\mathrm{in}}_t$ in the same way. Concretely, in the definition of $\mathbf{c}'_j$ in Subsection 4.2, we replace the summands with $r = 1$ by the single gadget product $\mathbf{c}\mathbf{G}^{-1}\big((\mathbf{C}_k)_{1,j}\mathbf{1}_1^\top\big)$, which carries the same payload $\mu(\mathbf{C}_k)_{1,j}(\mathbf{s}_k)_1$ under the public matrix $\mathbf{A}\mathbf{G}^{-1}\big((\mathbf{C}_k)_{1,j}\mathbf{1}_1^\top\big)$, and replace the corresponding summands of $\mathbf{A}'_j$ by this matrix. Key generation computes it from mpk as before, and its error is at most $\tau n\,\mathrm{err}(\mathbf{c})$, since $\mathbf{G}^{-1}\big((\mathbf{C}_k)_{1,j}\mathbf{1}_1^\top\big)$ has only τ nonzero rows; this is the part of the bound $\tau(2\tau n\beta_{\mathsf{RHS}} + n\,\mathrm{err}(\mathbf{c}))$ for the replaced summands that comes from their terms $d_{1,j,i}\mathbf{c}$, so that the new bound is smaller. The ciphertext thus omits the digit encodings $\mathbf{c}^{\mathrm{aut}}_{k,1,j,i}$ and $\mathbf{c}^{\mathrm{in}}_{t,1,j,i}$ of the first rows, which halves the RHS data of every gadget ciphertext from $8\tau^3$ to $4\tau^3$ ring elements; since this only removes components from the ciphertext, Theorem 2 carries over.

Compressed digit encodings. The RHS data dominate the ciphertext, and each digit encoding has 2τ entries. We transmit each digit encoding in compressed form, with two entries over a larger modulus, and let the evaluator decompress it by modulus switching [BV11, BGV12]. Let $Q := pq$ for a modulus $p \approx q$ coprime to q. Encryption computes $\check{\mathbf{c}} := \mathbf{s}'(\check{\mathbf{A}} - p\,d\,\mathbf{I}_2) + \check{\mathbf{e}} \in \mathcal{R}_Q^{1\times 2}$ for a public $\check{\mathbf{A}} \in \mathcal{R}_Q^{2\times 2}$, and the evaluator computes

$$\lfloor \check{\mathbf{c}}\mathbf{G}/p \rceil \bmod q \;:\llbracket d\mathbf{s}' \rrbracket_{\mathbf{s}'}^{\lfloor \check{\mathbf{A}}\mathbf{G}/p \rceil \bmod q},$$

whose error is at most $(q/p)\|\check{\mathbf{e}}\|_\infty + n\|\bar{s}'\|_\infty/2 + 1$, since the secret is short. This reduces the RHS data of a gadget ciphertext from $8\tau^3$ elements of $\mathcal{R}_q$ to $8\tau^2$ elements of $\mathcal{R}_Q$, i.e., by a factor of

about $\tau/2$ for the same ring dimension, at the cost of a larger n for the modulus Q. For the security proof, the anchor $(\mathbf{b}, \mathbf{B})$ is defined modulo Q, from which rounding by p recovers it modulo q, and the secret key grows by $W(m_c\tau + 1)$ short ring elements, where $m_c \leq m_B$ is the number of columns of $\mathbf{B}$ that the proof uses to program the compressed encodings. Appendix B gives the resulting scheme and proves its security under circular Ring-LWE modulo Q.

Trapdoor and preimage sampling. We use the statistical instantiation of the trapdoor (Lemma 4). Preimage sampling is based on the implementation of Datta et al. [DRTV26], which follows Peikert's offline/online sampler [Pei10]: the perturbation is drawn as a continuous Gaussian from a Cholesky factor of its covariance, precomputed at setup, and then rounded to the integers; for the rounding, we use the randomized rounding of [Pei10]. The gadget part is sampled as in [CGM+18]. The following heuristics remain.

- *Floating-point arithmetic.* The Cholesky factor and the perturbations are computed in double precision, and the effect of rounding errors on the output distribution is not analyzed.
- *Ideal samplers.* The statistical distances of the samplers from their ideal distributions, including the randomized rounding and the gadget sampling, are not included in the statistical budget of Subsection 6.2. With our widths, the convolution and gadget-sampling theorems bound these distances far below the budget; we only omit them from it.
- *Truncation.* Discrete Gaussian samples are drawn by rejection sampling with a fixed tail cut, and the coordinates of a key are bounded by 6.5 times its width, a constant chosen experimentally.

6.2 Parameter Selection

Following Datta et al. [DRTV26, Section 6.1], we keep every condition of Subsection 5.1 on the distributions of the scheme itself, relax the conditions that only the security reduction (Appendix A) uses, and estimate the concrete security of the underlying Ring-LWE instance separately. These relaxations change only the widths of the distributions, not the algorithms, but the resulting parameters are not covered by Theorem 2.

Conditions we relax. We drop noise flooding, which hides terms that appear only in the components computed by the reduction in H_2, and sample the errors of fresh encodings and of the attribute symbols from χ, i.e., $\sigma_{\mathrm{fl}} = \sigma_\chi$. We also drop the term $Nn\, B_{\widehat{\mathbf{R}}}$ of the key width, which serves only the preimage sampling with the gadget trapdoor of Lemma 11 in H_3, and set σ from $s_1(\mathbf{T})$ alone, under which $\mathsf{SampleLeftBlock}$ still outputs $D_{\Lambda_{\mathbf{y}}(\mathbf{M}_F),\sigma}$ (Lemma 6). We also drop the uniformity conditions on σ_Z and $\sigma_{\mathbf{R}}$, which concern the matrices $\mathbf{Z}_t$ and $\mathbf{R}$ that appear only in the hybrids, so that $m_B = \bar{m} + \tau$ is determined by the trapdoor alone. As in [DRTV26, Section 6.1], we set the factors $\omega(\sqrt{\log n})$ in the Gaussian widths and the constants hidden by $O(\cdot)$ to 1.

Security and correctness targets. We choose the ring dimension n, the modulus q, the gadget base β_{g}, and the radix B so that the Ring-LWE instance of the encryption secret, with the helpers and the attribute symbols as additional samples, has at least 128 bits of security according to the lattice estimator [APS15], and that $\mathbf{B}$ is within statistical distance 2^{-30} from uniform by the leftover hash lemma, i.e., the uniformity condition on $\bar{m}$ and $\sigma_{\mathbf{T}}$ in Subsection 5.1 holds for this distance. For correctness, we require average-case rather than worst-case correctness: decryption fails with probability less than 2^{-128} over the randomness of key generation and encryption. We prove in Lean that this bound on the failure probability holds for the concrete parameter sets used in our

experiments. The proved statement is not written by hand: mxx derives it deterministically from the definition of the protocol in its domain-specific language.

7 Evaluation

We measure the concrete efficiency of our implementation on a heterogeneous predicate and compare it with a baseline for PE schemes that evaluate FHE over BGG+ encodings [GVW15, BTVW17, WWLW25] (Subsection 7.1). Results on the concrete efficiency for branching programs of different scales will be added in a future version.

Experimental setup. All experiments run on a single NVIDIA A100 SXM GPU with 80 GB of memory, on a host with 16 cores of an AMD EPYC 7742 CPU and 250 GB of RAM, with parameters selected as in Subsection 6.2. Since the data of the scheme, such as the ciphertext, do not fit into the memory of the GPU, every algorithm streams its data between disk and GPU memory, and the reported running times include these transfers. We report the modulus size in bits, the ring dimension, the sizes of mpk and ct, and the running times of KeyGen, Enc, and Dec.

7.1 Concrete Efficiency on a Heterogeneous Predicate

Measured predicate. We evaluate a heterogeneous key-policy predicate that imposes both a format and a range on the same attribute. The attribute is a calendar date given as two symbols, its month and its day, each encoded in 5 bits; thus the attribute has $\ell = 2$ symbols and 10 bits, and $|\Sigma| = 32$. The key specifies a date window $[a, b]$ for valid dates $a \leq b$ of the same year, and the predicate accepts if and only if both of the following hold:

- *Format:* the date is valid, i.e., its month is between 1 and 12 and its day between 1 and the number of days of that month, with 29 days for February;
- *Range:* the date lies in $[a, b]$ in the order of months and then days.

The valid dates form a regular language over Σ, and the two conditions are independent: for instance, February 30 lies in the window from January 15 to March 15 but is not a valid date.

Branching program. A single branching program of length $T = 2$ checks both conditions, reading the month and then the day. After the month, its state records two comparisons: whether the month equals that of a or is already greater (EQ or GT), and whether it equals that of b or is already smaller (EQ or LT). Once the month is smaller than that of a or greater than that of b, or is not between 1 and 12, the program moves to a rejecting state, which it never leaves. To check the day, the program needs the number of days of the month. If the month equals that of a or of b, the key determines it; otherwise, the state also records whether the month has 31, 30, or 29 days. The program thus has width $w = 7$: the three states in which the month equals that of a or of b, the three states for a month strictly inside the window, and the rejecting state. After padding w to 8, its lookup width is $W = 8 \cdot 32 = 256$.

Baseline. As a baseline, we consider a single Ring-GSW multiplication simulated as a Boolean circuit over BGG+ encodings, the basic step of PE schemes that evaluate FHE over BGG+ encodings [GVW15, BTVW17, WWLW25]. The nonlinear predicate considered here requires at least one multiplication in this straightforward FHE implementation. The baseline therefore lower-bounds the cost of this implementation on our predicate; this claim does not extend to optimized implementations.

The circuit takes two Ring-GSW ciphertexts $\mathbf{C}_1, \mathbf{C}_2 \in \mathcal{R}_{q_G}^{2\times 2\ell_G}$ of ring dimension n_G and prime modulus q_G, with the binary gadget of length $\ell_G = \lceil \log_2 q_G \rceil$, and outputs $\mathbf{C}_1\mathbf{G}^{-1}(\mathbf{C}_2)$. Every coefficient is given by its ℓ_G bits in Montgomery form, so the circuit has $8n_G\ell_G^2$ input bits and uses only AND, XOR, and NOT gates. Apart from converting the entries of $\mathbf{C}_2$ out of Montgomery form, it performs no modular multiplication: the digits of $\mathbf{G}^{-1}(\mathbf{C}_2)$ are the bits of these entries, so each entry of the product is a sum of $2\ell_G$ negacyclic products of an entry of $\mathbf{C}_1$ with a binary polynomial, whose partial products are AND gates and which remains in Montgomery form. The circuit sums these products exactly over the integers in carry-save form, by full adders that never propagate carries, and merges each output coefficient into one integer with a Kogge–Stone adder before reducing it modulo q_G; the signs of the negacyclic wrap-around are NOT gates and a public offset. The circuit therefore has $\Theta(n_G^2\ell_G^3)$ gates, and its depth grows logarithmically in n_G and ℓ_G.

With more multiplications, the Ring-GSW noise requires a modulus with several primes, and the circuit works modulo each prime and recombines the results by the CRT as a linear combination over the encodings, which is correct only if the encoding modulus is a multiple of the Ring-GSW modulus. We therefore take q_G to be one of the 44-bit primes of the encoding modulus, and $n_G = 2048$, the smallest ring dimension for which it reaches 128-bit security; a Ring-GSW modulus with several primes would need a larger n_G, so this choice underestimates the cost of the baseline.

We choose the smallest $\log_2 q$ and n for which this circuit decrypts correctly at the same security level as our scheme. These parameters are too large to run the circuit on actual hardware. Instead, as for our scheme (Subsection 6.2), we prove in Lean that, for these parameters, the noise of the encoding obtained after evaluating the circuit stays below the decryption threshold except with probability less than 2^{-128}.

Evaluation methods. For both schemes, we report the logical bit-packed sizes of mpk and of the ciphertext per attribute bit, counting each coefficient according to its modulus: $\lceil \log_2 q \rceil$ bits modulo q and $\lceil \log_2 Q \rceil$ bits modulo Q. In our scheme, the compressed digit encodings and the anchor are stored modulo $Q = pq$; as recalled in Subsection 6.1, decompression switches to q before encoding evaluation, so the evaluation modulus has approximately half the bit length of Q, since $p \approx q$. For both schemes, public-key sizes count the matrices in expanded form. For our scheme, we count each distinct matrix in mpk as defined in Subsection 5.1 once, in the representation obtained after applying the optimizations of Subsection 6.1. Here expansion means generating the matrices from the seed; compact digit matrices remain in their compressed form. In the implementation, mpk stores the first row $\mathbf{B}^{(1)}$ explicitly and contains a fixed-size public seed from which all remaining public-key matrix entries are derived using a hash function (Subsection 6.1); Table 1 counts the expanded matrix data rather than this seed representation. For the 10-bit attribute, our expanded public key occupies 676,453,788,896 bytes, or 67.65 GB per attribute bit after rounding up to two decimal places, with 1 GB defined as 10^9 bytes. In both schemes, the ciphertext is dominated by GSW-form ciphertexts of the input and the BGG+ encodings of their digits. For our scheme, we also measure the running times of KeyGen, Enc, and Dec in the setup described at the beginning of this section.

Since the baseline is implemented but cannot be run end to end at its parameters, we compare the decryption of both schemes by the numbers of 64-bit integer multiplications and digit decompositions, which we count in the same way for both. We count these operations without running decryption, since they depend only on the parameters and on the operations that decryption performs on ring elements. A ring element modulo q is stored as one polynomial of n coefficients modulo each prime of the RNS basis, and every operation on ring elements, such as a multiplication, an NTT, or a gadget decomposition, acts on these polynomials separately. For every such operation of decryption,

we take the number of 64-bit operations that it performs on one polynomial and multiply it by the number of primes; for instance, a product of two ring elements in NTT form costs n modular multiplications per prime, and an NTT costs about $\frac{n}{2}\log_2 n$ of them. We then add these numbers over all operations of decryption, where the operations are counted as follows:

- "Mul" counts 64-bit integer multiplications: a modular multiplication by a precomputed constant, as in the NTT, takes three of them by Shoup's method, and a $64 \times 64 \to 128$-bit product or a general 64-bit modular multiplication takes one;
- "Decomp." counts the 64-bit RNS residues that are decomposed into gadget digits, i.e., one for each coefficient of each ring element and each prime of the RNS basis that the decomposition uses.

Results. Table 1 compares the baseline with our scheme on this predicate. Even though the baseline is a single Ring-GSW multiplication and our scheme evaluates the whole predicate, our scheme is smaller and cheaper in every measure. Its modulus is about 12 times shorter and its ring dimension 16 times smaller, so that every ring element is about 190 times smaller. The sizes per attribute bit shrink by far more, by about 2.06×10^9 for the expanded mpk and 2.04×10^9 for the ciphertext, since our scheme also needs far fewer ring elements per attribute bit: the baseline encodes every bit of a GSW ciphertext of each attribute bit, whereas our scheme encodes only the digits of one gadget ciphertext per symbol. Decryption performs about 2.67×10^{16} times fewer 64-bit integer multiplications and 1.86×10^{14} times fewer digit decompositions. This gap is much larger than that in the sizes, since the baseline evaluates a circuit with $\Theta(n_{\mathrm{G}}^2 \ell_{\mathrm{G}}^3)$ gates, each of which costs operations on encodings under its larger parameters, whereas our decryption consists of $T = 2$ steps, each of one asymmetric multiplication and one lookup table of width $W = 256$.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Gilbert [Gil10], pages 553–572. doi:10.1007/978-3-642-13190-5_28.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 21–40. Springer, Berlin, Heidelberg, December 2011. doi:10.1007/978-3-642-25385-0_2.

[AGM+13] Joseph A. Akinyele, Christina Garman, Ian Miers, Matthew W. Pagano, Michael Rushanan, Matthew Green, and Aviel D. Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, June 2013. doi:10.1007/s13389-013-0057-3.

[Agr17] Shweta Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 3–35. Springer, Cham, August 2017. doi:10.1007/978-3-319-63688-7_1.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[AR17] Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In Kalai and Reyzin [KR17], pages 173–205. doi:10.1007/978-3-319-70500-2_7.

[Ban93] Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–635, 1993.

[Bar89] David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Berlin, Heidelberg, May 2014. doi:10.1007/978-3-642-55220-5_30.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539. Springer, Berlin, Heidelberg, August 2016. doi:10.1007/978-3-662-53018-4_19.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012. doi:10.1145/2090236.2090262.

[BKS19] Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 3–33. Springer, Cham, May 2019. doi:10.1007/978-3-030-17656-3_1.

[BOC92] Michael Ben-Or and Richard Cleve. Computing algebraic formulas using a constant number of registers. *SIAM Journal on Computing*, 21(1):54–58, 1992.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Berlin, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_50.

[BRF13] Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors. *45th ACM STOC*. ACM Press, June 2013.

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In Kalai and Reyzin [KR17], pages 264–302. doi:10.1007/978-3-319-70500-2_10.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011. doi:10.1109/FOCS.2011.12.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 535–554. Springer, Berlin, Heidelberg, February 2007. doi:10.1007/978-3-540-70936-7_29.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon

and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Berlin, Heidelberg, December 2016. doi:10.1007/978-3-662-53887-6_1.

[CGM+18] Cheng Chen, Nicholas Genise, Daniele Micciancio, Yuriy Polyakov, and Kurt Rohloff. Implementing token-based obfuscation under (ring) LWE. Cryptology ePrint Archive, Report 2018/1222, 2018. URL: https://eprint.iacr.org/2018/1222.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Gilbert [Gil10], pages 523–552. doi:10.1007/978-3-642-13190-5_27.

[CJP21] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In *Cyber Security Cryptography and Machine Learning (CSCML 2021)*, volume 12716 of *LNCS*, pages 1–19. Springer, 2021. doi:10.1007/978-3-030-78086-9_1.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 409–437. Springer, Cham, December 2017. doi:10.1007/978-3-319-70694-8_15.

[DDP+18] Wei Dai, Yarkın Doröz, Yuriy Polyakov, Kurt Rohloff, Hadi Sajjadpour, Erkay Savaş, and Berk Sunar. Implementation and evaluation of a lattice-based key-policy ABE scheme. *IEEE Transactions on Information Forensics and Security*, 13(5):1169–1184, 2018.

[DRTV26] Pratish Datta, Yannis Rouselakis, Junichi Tomida, and Nikhil Vanjani. Large-universe (multi-authority) ABE from LWE. Cryptology ePrint Archive, Report 2026/1895, 2026. URL: https://eprint.iacr.org/2026/1895.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. URL: https://eprint.iacr.org/2012/144.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 498–527. Springer, Berlin, Heidelberg, March 2015. doi:10.1007/978-3-662-46497-7_20.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In Pointcheval and Johansson [PJ12], pages 465–482. doi:10.1007/978-3-642-29011-4_28.

[Gil10] Henri Gilbert, editor. *EUROCRYPT 2010*, volume 6110 of *LNCS*. Springer, Berlin, Heidelberg, May / June 2010.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Boneh et al. [BRF13], pages 555–564. doi:10.1145/2488608.2488678.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Umans [Uma17], pages 612–621. doi:10.1109/FOCS.2017.62.

[GMW15] Romain Gay, Pierrick Méaux, and Hoeteck Wee. Predicate encryption for multi-dimensional range queries from lattices. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 752–776. Springer, Berlin, Heidelberg, March / April 2015. doi:10.1007/978-3-662-46447-2_34.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. doi:10.1145/1374376.1374407.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Berlin, Heidelberg, August 2013. doi:10.1007/978-3-642-40041-4_5.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Boneh et al. [BRF13], pages 545–554. doi:10.1145/2488608.2488677.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Berlin, Heidelberg, August 2015. doi:10.1007/978-3-662-48000-7_25.

[HLL23] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *64th FOCS*, pages 415–434. IEEE Computer Society Press, November 2023. doi:10.1109/FOCS57990.2023.00031.

[HS14] Shai Halevi and Victor Shoup. Algorithms in HElib. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 554–571. Springer, Berlin, Heidelberg, August 2014. doi:10.1007/978-3-662-44371-2_31.

[IP08] Vincenzo Iovino and Giuseppe Persiano. Hidden-vector encryption with groups of prime order. In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 75–88. Springer, Berlin, Heidelberg, September 2008. doi:10.1007/978-3-540-85538-5_5.

[JLWG25] Haoxiang Jin, Feng-Hao Liu, Zhedong Wang, and Dawu Gu. Discrete gaussians modulo sub-lattices: New leftover hash lemmas for discrete gaussians. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part II*, volume 15675 of *LNCS*, pages 301–330. Springer, Cham, May 2025. doi:10.1007/978-3-031-91823-0_10.

[Kat17] Shuichi Katsumata. On the untapped potential of encoding predicates by arithmetic circuits and their applications. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 95–125. Springer, Cham, December 2017. doi:10.1007/978-3-319-70700-6_4.

[KLLL26] Jaehyung Kim, Hanjun Li, Huijia Lin, and Zeyu Liu. SIMD HSS and aHMAC from interval encoding with application to one-bit-per-gate garbling. In *Advances in Cryptology – CRYPTO 2026*, pages 658–691. Springer, 2026.

[KR17] Yael Kalai and Leonid Reyzin, editors. *TCC 2017, Part I*, volume 10677 of *LNCS*. Springer, Cham, November 2017.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Berlin, Heidelberg, April 2008. doi:10.1007/978-3-540-78967-3_9.

[LLW21] Qiqi Lai, Feng-Hao Liu, and Zhedong Wang. New lattice two-stage sampling technique and its applications to functional encryption - stronger security and smaller ciphertexts. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 498–527. Springer, Cham, October 2021. doi:10.1007/978-3-030-77870-5_18.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Gilbert [Gil10], pages 1–23. doi:10.1007/978-3-642-13190-5_1.

[Mac26] Machina iO, Ethereum Foundation. mxx: Write a lattice-based cryptographic protocol in a Rust-based DSL, much as you would write it on paper, and run it on GPUs, with no CUDA and no hand-tuning of GPU memory or parallelism. https://github.com/MachinaIO/mxx, 2026.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In Pointcheval and Johansson [PJ12], pages 700–718. doi:10.1007/978-3-642-29011-4_41.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

[Par11] Jong Hwan Park. Inner-product encryption under standard assumptions. *DCC*, 58(3):235–257, 2011. doi:10.1007/s10623-010-9405-9.

[Pei10] Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 80–97. Springer, Berlin, Heidelberg, August 2010. doi:10.1007/978-3-642-14623-7_5.

[PJ12] David Pointcheval and Thomas Johansson, editors. *EUROCRYPT 2012*, volume 7237 of *LNCS*. Springer, Berlin, Heidelberg, April 2012.

[Sco11] Michael Scott. On the efficient implementation of pairing-based protocols. In Liqun Chen, editor, *13th IMA International Conference on Cryptography and Coding*, volume 7089 of *LNCS*, pages 296–308. Springer, Berlin, Heidelberg, December 2011. doi:10.1007/978-3-642-25516-8_18.

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 535–559. Springer, Cham, August 2022. doi:10.1007/978-3-031-15802-5_19.

[Uma17] Chris Umans, editor. *58th FOCS*. IEEE Computer Society Press, October 2017.

[Wee22] Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 217–241. Springer, Cham, May / June 2022. doi:10.1007/978-3-031-07085-3_8.

[WWLW25] Yuejun Wang, Baocang Wang, Qiqi Lai, and Huaxiong Wang. Predicate encryption from lattices: Enhanced compactness and refined functionality. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part III*, volume 15676 of *LNCS*, pages 37–67. Springer, Cham, May 2025. doi:10.1007/978-3-031-91826-1_2.

[WWW22] Brent Waters, Hoeteck Wee, and David J. Wu. Multi-authority ABE from lattices without random oracles. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 651–679. Springer, Cham, November 2022. doi:10.1007/978-3-031-22318-1_23.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Umans [Uma17], pages 600–611. doi:10.1109/FOCS.2017.61.

A Proof of Theorem 2

Parameter conditions. Let $q = \prod_{i=1}^{L_q} q_i$ for distinct primes $q_i \equiv 1 \pmod{2n}$, and let $m_B = \bar{m}+\tau$. Use Gaussian widths σ_χ for $\chi := D_{\mathcal{R},\sigma_\chi}$, σ_{fl} for fresh-encoding errors $\chi_{\mathsf{fl}} := D_{\mathcal{R},\sigma_{\mathsf{fl}}}$, $\sigma_{\mathbf{T}}$ for the trapdoor, and σ for keys; $\sigma_{\mathbf{R}}$ and σ_Z are auxiliary widths used only in the proof. Require $\sigma_\chi \geq \omega(\sqrt{\log n})$ and $\sigma_{\mathbf{R}}, \sigma_Z, \sigma_{\mathbf{T}}$ in the range of Lemma 2. Use the error bounds B_χ and B_F defined in Subsection 5.2, and set $\varrho := L_q \log(2n) + 2\lambda$. For the simulation matrices, set

$$D := 2\tau n, \qquad R_* := \max\{1, \sigma_{\mathbf{R}}\sqrt{\lambda}\}, \qquad Z_* := \max\{1, \sigma_Z\sqrt{\lambda}\}, \qquad \gamma := WD^{b+2},$$

$$\Delta_{\mathrm{sim}} := WD^{b+1}\big(2\tau D R_* + Z_* + 4\tau R_* + 4\big), \qquad B_{\widehat{\mathbf{R}}} := \gamma^T(R_* + \Delta_{\mathrm{sim}}). \tag{5}$$

The coefficient bound $B_{\widehat{\mathbf{R}}}$ is justified after Corollary 1 below.

We require:

1. *Hardness:* circular Ring-LWE (Assumption 1) is hard for (n, q, χ_s, χ);
2. *Uniformity:*

$$\bar{m}(\log \sigma_{\mathbf{T}} - 1) \geq \log q + \varrho, \quad m_B(\log \sigma_Z - 1) \geq 2\log q + \varrho, \quad m_B(\log \sigma_{\mathbf{R}} - 1) \geq 3\log q + \varrho,$$

   as required by Lemma 2;
3. *Flooding:* $\sigma_{\mathsf{fl}} \geq \lambda^{\omega(1)} \cdot \max(B_{e\mathbf{R}}, B_{e\mathbf{Z}})$ for $B_{e\mathbf{R}} := \sqrt{m_B n}\, B_\chi \sigma_{\mathbf{R}}\sqrt{\lambda}$ and $B_{e\mathbf{Z}} := \sqrt{m_B n}\, B_\chi \sigma_Z \sqrt{\lambda}$;
4. *Key width:* $\sigma \geq \omega(\sqrt{\log n}) \cdot \max\big(s_1(\mathbf{T}),\, (2\tau n)^b N n\, B_{\widehat{\mathbf{R}}}\big)$;
5. *Correctness:* $B_\chi + Nn\sigma\sqrt{\lambda} \cdot \max(B_\chi, B_F) < \lfloor q/4 \rceil$.

Here N is the key length defined in Subsection 5.1. The bounds $B_{e\mathbf{R}}$ and $B_{e\mathbf{Z}}$ follow from Lemma 1 applied to the Gaussian products $\sigma_i(\mathbf{e}_b)\mathbf{R}$ and $\mathbf{e}_b\mathbf{Z}_t$. The bounds B_F and $B_{\widehat{\mathbf{R}}}$ have the same per-step factor γ, allowing $\log_2 q = O(\lambda + (T+1)\log_2 \gamma)$. Concrete parameters are selected in Section 6.

Proof strategy. Let $\mathcal{A}$ be a PPT adversary and $\beta \in \{0,1\}$. We define a sequence of hybrids $\mathsf{H}_0, \ldots, \mathsf{H}_5$, where H_0 is $\mathsf{Exp}^{\mathrm{wAH}}_{\mathcal{A},\beta}$ and H_5 does not depend on β; consecutive hybrids are indistinguishable, which proves the theorem. Throughout, let $x^* := x_\beta$ and $m^* := m_\beta$ be the challenge attribute and message; by the definition of the key oracle, every queried function satisfies $F(x^*) = 1$.

Before defining the hybrids, we develop the tool that allows the reduction to answer key queries without the trapdoor of $\mathbf{B}^{(1)}$: a *simulation form* that expresses every public matrix produced by the evaluation through the matrix $\mathbf{L}$ of (3) and a short matrix. Recall from Section 3 that the rows of matrices with W rows are indexed by $\mathcal{H}$, that $\boldsymbol{\delta}_i \in \{0,1\}^{1\times W}$ is the unit vector with 1 in the position indexed by i, and that $\boldsymbol{\delta}_i^\top \otimes \mathbf{a}$ places a row $\mathbf{a}$ in the row indexed by i and is zero elsewhere. Let $\bar{\mathbf{s}}_{\mathcal{H}} = (\sigma_i(\bar{s}))_{i\in\mathcal{H}}$. For $i \in \mathcal{H}$ and $\mathbf{A}$ with two rows, $\mathbf{s}_i\mathbf{A} = \bar{\mathbf{s}}_{\mathcal{H}}(\boldsymbol{\delta}_i^\top \otimes \mathbf{A}^{(1)}) + \mathbf{A}^{(2)}$. Recall from key generation that

$$\mathbf{L} := \left[\ \big(\boldsymbol{\delta}_i^\top \otimes \sigma_i(\mathbf{B}^{(1)})\big)_{i\in\mathcal{H}} \ \middle|\ \big(\boldsymbol{\delta}_i^\top \otimes \sigma_i(\mathbf{H}_k^{(1)}) - \boldsymbol{\delta}_{i\cdot k}^\top \otimes \mathbf{G}^{(1)}\big)_{i\in\mathcal{H},\, k\in\mathcal{J}}\ \right] \in \mathcal{R}_q^{W\times N_L},$$

with $N_L = Wm_B + 2\tau W|\mathcal{J}|$. We label its column blocks β_i for the block $\boldsymbol{\delta}_i^\top \otimes \sigma_i(\mathbf{B}^{(1)})$ of width m_B, and $\eta_{i,k}$ for the block $\boldsymbol{\delta}_i^\top \otimes \sigma_i(\mathbf{H}_k^{(1)}) - \boldsymbol{\delta}_{i\cdot k}^\top \otimes \mathbf{G}^{(1)}$ of width 2τ. For a column block θ of $\mathbf{L}$, labeled β_i or $\eta_{i,k}$, let $\mathbf{J}_\theta$ be the 0/1 matrix that embeds a block into the columns of θ, so that

$$\mathbf{L}\mathbf{J}_{\beta_i} = \boldsymbol{\delta}_i^\top \otimes \sigma_i(\mathbf{B}^{(1)}), \qquad \mathbf{L}\mathbf{J}_{\eta_{i,k}} = \boldsymbol{\delta}_i^\top \otimes \sigma_i(\mathbf{H}_k^{(1)}) - \boldsymbol{\delta}_{i\cdot k}^\top \otimes \mathbf{G}^{(1)}.$$

Definition 3 (Simulation form)**.** A declaration $\mathbf{c} : [\![\mu \mathbf{s}_{i'}]\!]^{\mathbf{A}}_{\mathbf{s}_i}$ with $i, i' \in \mathcal{H}$ and $\mu \in \mathcal{R}_q$ is in *simulation form* with respect to a short matrix $\widehat{\mathbf{R}}$ if

$$\boldsymbol{\delta}_i^\top \otimes \mathbf{A}^{(1)} - \boldsymbol{\delta}_{i'}^\top \otimes \mu \mathbf{G}^{(1)} = \mathbf{L}\widehat{\mathbf{R}}. \tag{6}$$

Similarly, $\mathbf{C} : \mathsf{GC}_{\mathbf{s}_i}(\nu \mathbf{s}_{i'})$ is in simulation form with respect to $\widehat{\mathbf{Z}}$ if $\boldsymbol{\delta}_i^\top \otimes \mathbf{C}^{(1)} - \boldsymbol{\delta}_{i'}^\top \otimes \nu \mathbf{G}^{(1)} = \mathbf{L}\widehat{\mathbf{Z}}$.

Simulation forms are used only by the challenger of the hybrids, which knows the challenge attribute x^* and hence every value μ and ν that appears in the evaluation; the values remain hidden from the adversary.

The left-hand side of (6) is the part of $\mathbf{s}_i\mathbf{A} - \mu \mathbf{s}_{i'}\mathbf{G}$ that depends on the secret:

$$\mathbf{s}_i\mathbf{A} - \mu \mathbf{s}_{i'}\mathbf{G} = \bar{\mathbf{s}}_{\mathcal{H}}\big(\boldsymbol{\delta}_i^\top \otimes \mathbf{A}^{(1)} - \boldsymbol{\delta}_{i'}^\top \otimes \mu \mathbf{G}^{(1)}\big) + \mathbf{A}^{(2)} - \mu \mathbf{G}^{(2)}.$$

Lemma 10 (Propagation of simulation forms)**.** *Let the inputs of a primitive operation of Section 4 be in simulation form, and let every encoding multiplied by an RHS ciphertext have $\|\mu\|_1 \le 1$. Then the output is in simulation form with respect to a matrix computable from those of the inputs:*

1. Add*: for inputs with respect to $\widehat{\mathbf{R}}_1$ and $\widehat{\mathbf{R}}_2$, the output is in simulation form with respect to $\widehat{\mathbf{R}}_1 + \widehat{\mathbf{R}}_2$.*
2. SMul_α*: for an input with respect to $\widehat{\mathbf{R}}$, the output, with payload $\alpha\mu$, is in simulation form with respect to $\widehat{\mathbf{R}}\mathbf{G}^{-1}(\alpha\mathbf{G})$.*
3. Mul*: let $\mathbf{c} : [\![\mu \mathbf{s}_i]\!]^{\mathbf{A}}_{\mathbf{s}_i}$ be in simulation form with respect to $\widehat{\mathbf{R}}$, and let the RHS data $\mathsf{RHS}_{\mathbf{s}_i}(\mathbf{C})$ consist of $\mathbf{C} : \mathsf{GC}_{\mathbf{s}_i}(\nu \mathbf{s}_{i'})$ with respect to $\widehat{\mathbf{Z}}$ and digit encodings $\mathbf{c}_{r,j,i''} : [\![d_{r,j,i''}\mathbf{s}_i]\!]^{\mathbf{A}_{r,j,i''}}_{\mathbf{s}_i}$ with respect to $\widehat{\mathbf{R}}_{r,j,i''}$. Then $\mathsf{Mul}(\mathbf{c}, \mathsf{RHS}_{\mathbf{s}_i}(\mathbf{C})) : [\![\mu\nu \mathbf{s}_{i'}]\!]^{\mathbf{A}'}_{\mathbf{s}_i}$ is in simulation form with respect to $\mathbf{S} + \widehat{\mathbf{R}}\mathbf{G}^{-1}(\mathbf{C}) + \mu\widehat{\mathbf{Z}}$, where $\mathbf{S} := \big(\sum_{r,i''} \widehat{\mathbf{R}}_{r,j,i''}\mathbf{G}^{-1}(\mathbf{A})\mathbf{G}^{-1}(2^{i''}\mathbf{1}_r^\top)\big)_j$.*
4. Aut_k *for $k \in \mathcal{K}$: let $\mathbf{c} : [\![\mu \mathbf{s}]\!]^{\mathbf{A}}_{\mathbf{s}}$ be in simulation form with respect to $\widehat{\mathbf{R}}$, and let the helper data of σ_k be in simulation form. Then $\mathsf{Aut}_k(\mathbf{c}) : [\![\sigma_k(\mu)\mathbf{s}]\!]^{\mathbf{H}_k\mathbf{G}^{-1}(\mathbf{A}'_k)}_{\mathbf{s}}$ is in simulation form with respect to a matrix computable from $\widehat{\mathbf{R}}$ and those of the helper data.*

Proof. We treat the four operations in turn.

Add *and* SMul_α. For Add, the identities (6) of the two inputs add up. For SMul_α, multiply (6) on the right by $\mathbf{G}^{-1}(\alpha\mathbf{G})$ and use $\mathbf{G}^{(1)}\mathbf{G}^{-1}(\alpha\mathbf{G}) = \alpha\mathbf{G}^{(1)}$.

Mul. We use two facts about the gadget. First, $(\boldsymbol{\delta}_i^\top \otimes \mathbf{X})\mathbf{Y} = \boldsymbol{\delta}_i^\top \otimes (\mathbf{X}\mathbf{Y})$ for every row $\mathbf{X}$ and matrix $\mathbf{Y}$ of matching sizes, and $\mathbf{G}^{(1)}\mathbf{G}^{-1}(\mathbf{M}) = \mathbf{M}^{(1)}$ for every $\mathbf{M}$ with two rows, since $\mathbf{G}\mathbf{G}^{-1}(\mathbf{M}) = \mathbf{M}$. Second, index the rows of $\mathbf{G}^{-1}(\cdot)$ by pairs $(r, i'') \in [2] \times [0, \tau - 1]$, so that the row (r, i'') holds the i''-th binary digit of the r-th entry. Since $2^{i''}$ has the single binary digit 1 at position i'', the vector $\mathbf{u}_{r,i''} := \mathbf{G}^{-1}(2^{i''}\mathbf{1}_r^\top)$ is the unit vector at position (r, i''). Since the digits $d_{r,j,i''}$ are the binary digits of the entries $C_{r,j}$ of $\mathbf{C}$, it follows that

$$\textstyle\sum_{r,i''} d_{r,j,i''}\mathbf{u}_{r,i''} = \mathbf{G}^{-1}(\mathbf{C})_j,$$

the j-th column of $\mathbf{G}^{-1}(\mathbf{C})$.

We now compute the left-hand side of (6) for the output, column by column. By (6) for the digit encodings,

$$\boldsymbol{\delta}_i^\top \otimes \mathbf{A}^{(1)}_{r,j,i''} = \mathbf{L}\widehat{\mathbf{R}}_{r,j,i''} + d_{r,j,i''}\,\boldsymbol{\delta}_i^\top \otimes \mathbf{G}^{(1)}.$$

Multiplying both sides on the right by $\mathbf{G}^{-1}(\mathbf{A})\mathbf{u}_{r,i''}$ gives

$$\big(\boldsymbol{\delta}_i^\top \otimes \mathbf{A}^{(1)}_{r,j,i''}\big)\mathbf{G}^{-1}(\mathbf{A})\mathbf{u}_{r,i''} = \mathbf{L}\widehat{\mathbf{R}}_{r,j,i''}\mathbf{G}^{-1}(\mathbf{A})\mathbf{u}_{r,i''} + d_{r,j,i''}\big(\boldsymbol{\delta}_i^\top \otimes \mathbf{G}^{(1)}\big)\mathbf{G}^{-1}(\mathbf{A})\mathbf{u}_{r,i''}.$$

By the first fact, the left-hand side equals $\boldsymbol{\delta}_i^\top \otimes \mathbf{A}^{(1)}_{r,j,i''}\mathbf{G}^{-1}(\mathbf{A})\mathbf{u}_{r,i''}$, and the last term equals $d_{r,j,i''}\,\boldsymbol{\delta}_i^\top \otimes \mathbf{A}^{(1)}\mathbf{u}_{r,i''}$, since $\mathbf{G}^{(1)}\mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}^{(1)}$. Hence, for every (r, i''),

$$\boldsymbol{\delta}_i^\top \otimes \mathbf{A}^{(1)}_{r,j,i''}\mathbf{G}^{-1}(\mathbf{A})\mathbf{u}_{r,i''} = \mathbf{L}\widehat{\mathbf{R}}_{r,j,i''}\mathbf{G}^{-1}(\mathbf{A})\mathbf{u}_{r,i''} + d_{r,j,i''}\,\boldsymbol{\delta}_i^\top \otimes \mathbf{A}^{(1)}\mathbf{u}_{r,i''}.$$

We now sum over (r, i'') and treat the three terms separately. By the definition $\mathbf{A}'_j = \sum_{r,i''}\mathbf{A}_{r,j,i''}\mathbf{G}^{-1}(\mathbf{A})\mathbf{u}_{r,i''}$ of Mul, the left-hand sides sum to $\boldsymbol{\delta}_i^\top \otimes \mathbf{A}'^{(1)}_j$. By the definition of $\mathbf{S}$, the first terms on the right sum to $\mathbf{L}\mathbf{S}_j$, where $\mathbf{S}_j$ is the j-th column of $\mathbf{S}$. By the second fact, the last terms sum to $\boldsymbol{\delta}_i^\top \otimes \mathbf{A}^{(1)}\sum_{r,i''} d_{r,j,i''}\mathbf{u}_{r,i''} = \boldsymbol{\delta}_i^\top \otimes \mathbf{A}^{(1)}\mathbf{G}^{-1}(\mathbf{C})_j$. Therefore

$$\boldsymbol{\delta}_i^\top \otimes \mathbf{A}'^{(1)}_j = \mathbf{L}\mathbf{S}_j + \boldsymbol{\delta}_i^\top \otimes \mathbf{A}^{(1)}\mathbf{G}^{-1}(\mathbf{C})_j.$$

Collecting the columns, $\boldsymbol{\delta}_i^\top \otimes \mathbf{A}'^{(1)} = \mathbf{L}\mathbf{S} + \boldsymbol{\delta}_i^\top \otimes \mathbf{A}^{(1)}\mathbf{G}^{-1}(\mathbf{C})$. Next, by (6) for $\mathbf{c}$, with $i' = i$, multiplied on the right by $\mathbf{G}^{-1}(\mathbf{C})$,

$$\boldsymbol{\delta}_i^\top \otimes \mathbf{A}^{(1)}\mathbf{G}^{-1}(\mathbf{C}) = \mathbf{L}\widehat{\mathbf{R}}\mathbf{G}^{-1}(\mathbf{C}) + \mu\,\boldsymbol{\delta}_i^\top \otimes \mathbf{C}^{(1)}.$$

Finally, by the simulation form of $\mathbf{C}$, multiplied by μ,

$$\mu\,\boldsymbol{\delta}_i^\top \otimes \mathbf{C}^{(1)} = \mathbf{L}\mu\widehat{\mathbf{Z}} + \boldsymbol{\delta}_{i'}^\top \otimes \mu\nu\mathbf{G}^{(1)}.$$

Substituting the last two identities into the first gives

$$\boldsymbol{\delta}_i^\top \otimes \mathbf{A}'^{(1)} - \boldsymbol{\delta}_{i'}^\top \otimes \mu\nu\mathbf{G}^{(1)} = \mathbf{L}\big(\mathbf{S} + \widehat{\mathbf{R}}\mathbf{G}^{-1}(\mathbf{C}) + \mu\widehat{\mathbf{Z}}\big),$$

which is (6) for the output $\mathbf{c}' : [\![\mu\nu\mathbf{s}_{i'}]\!]^{\mathbf{A}'}_{\mathbf{s}_i}$.

Aut_k. We follow the three steps of Aut_k in Section 4. First, we apply σ_k entrywise. Let $\mathbf{P}_k$ be the row permutation $\boldsymbol{\delta}_j^\top \mapsto \boldsymbol{\delta}_{k\cdot j}^\top$, so that $\mathbf{P}_k(\boldsymbol{\delta}_j^\top \otimes \mathbf{x}) = \boldsymbol{\delta}_{k\cdot j}^\top \otimes \mathbf{x}$ for every row $\mathbf{x}$. We use three properties of σ_k: it is a ring automorphism, so that $\sigma_k(\mathbf{X}\mathbf{Y}) = \sigma_k(\mathbf{X})\sigma_k(\mathbf{Y})$ entrywise; it satisfies $\sigma_k \circ \sigma_j = \sigma_{k\cdot j}$; and it fixes integers, and hence the entries of $\boldsymbol{\delta}_j$ and of $\mathbf{G}^{(1)}$. We first show that $\mathbf{P}_k\sigma_k(\mathbf{L}) = \mathbf{L}\boldsymbol{\Pi}_k$ for the permutation $\boldsymbol{\Pi}_k$ of the column blocks of $\mathbf{L}$ that places the block $\beta_{k\cdot j}$ at the position of β_j and the block $\eta_{k\cdot j,k'}$ at the position of $\eta_{j,k'}$; since $j \mapsto k\cdot j$ is a bijection of $\mathcal{H}$, this is a permutation. For the block β_j, the three properties give

$$\mathbf{P}_k\sigma_k\big(\boldsymbol{\delta}_j^\top \otimes \sigma_j(\mathbf{B}^{(1)})\big) = \mathbf{P}_k\big(\boldsymbol{\delta}_j^\top \otimes \sigma_{k\cdot j}(\mathbf{B}^{(1)})\big) = \boldsymbol{\delta}_{k\cdot j}^\top \otimes \sigma_{k\cdot j}(\mathbf{B}^{(1)}),$$

which is the block $\beta_{k\cdot j}$ of $\mathbf{L}$. For the block $\eta_{j,k'}$, they give

$$\mathbf{P}_k\sigma_k\big(\boldsymbol{\delta}_j^\top \otimes \sigma_j(\mathbf{H}^{(1)}_{k'}) - \boldsymbol{\delta}_{j\cdot k'}^\top \otimes \mathbf{G}^{(1)}\big) = \boldsymbol{\delta}_{k\cdot j}^\top \otimes \sigma_{k\cdot j}(\mathbf{H}^{(1)}_{k'}) - \boldsymbol{\delta}_{k\cdot j\cdot k'}^\top \otimes \mathbf{G}^{(1)},$$

which is the block $\eta_{k\cdot j,k'}$ of $\mathbf{L}$. Now, the simulation form of $\mathbf{c}$ reads $\boldsymbol{\delta}_1^\top \otimes \mathbf{A}^{(1)} - \boldsymbol{\delta}_1^\top \otimes \mu\mathbf{G}^{(1)} = \mathbf{L}\widehat{\mathbf{R}}$. Applying σ_k entrywise to both sides gives $\boldsymbol{\delta}_1^\top \otimes \sigma_k(\mathbf{A})^{(1)} - \boldsymbol{\delta}_1^\top \otimes \sigma_k(\mu)\mathbf{G}^{(1)} = \sigma_k(\mathbf{L})\sigma_k(\widehat{\mathbf{R}})$. Multiplying both sides on the left by $\mathbf{P}_k$, which moves the row indexed by 1 to the row indexed by k, gives

$$\boldsymbol{\delta}_k^\top \otimes \sigma_k(\mathbf{A})^{(1)} - \boldsymbol{\delta}_k^\top \otimes \sigma_k(\mu)\mathbf{G}^{(1)} = \mathbf{P}_k\sigma_k(\mathbf{L})\sigma_k(\widehat{\mathbf{R}}) = \mathbf{L}\boldsymbol{\Pi}_k\sigma_k(\widehat{\mathbf{R}}).$$

Since $\sigma_k(\mathbf{s}) = \mathbf{s}_k$, the result $\sigma_k(\mathbf{c})$ satisfies $\sigma_k(\mathbf{c}) : [\![\sigma_k(\mu)\mathbf{s}_k]\!]^{\sigma_k(\mathbf{A})}_{\mathbf{s}_k}$, and the last identity is (6) with $i = i' = k$, with μ, $\mathbf{A}$, and $\widehat{\mathbf{R}}$ replaced by $\sigma_k(\mu)$, $\sigma_k(\mathbf{A})$, and $\boldsymbol{\Pi}_k\sigma_k(\widehat{\mathbf{R}})$; hence $\sigma_k(\mathbf{c})$ is in simulation form with respect to $\boldsymbol{\Pi}_k\sigma_k(\widehat{\mathbf{R}})$. Second, the case of Mul, applied to $\sigma_k(\mathbf{c})$ and $\mathsf{RHS}_{\mathbf{s}_k}(\mathbf{C}_k)$ with $i = k$ and $i' = 1$, shows that $\mathbf{r}_k : [\![\sigma_k(\mu)\mathbf{s}]\!]^{\mathbf{A}'_k}_{\mathbf{s}_k}$ is in simulation form with respect to a matrix $\widehat{\mathbf{R}}'_k$,

i.e., $\boldsymbol{\delta}_k^\top \otimes \mathbf{A}_k'^{(1)} - \boldsymbol{\delta}_1^\top \otimes \sigma_k(\mu)\mathbf{G}^{(1)} = \mathbf{L}\widehat{\mathbf{R}}_k'$. Third, the mask switch: the helper $\mathbf{h}_k : [\![\mathbf{s}_k]\!]_{\mathbf{s}}^{\mathbf{H}_k}$ is in simulation form with respect to $\mathbf{J}_{\eta_{1,k}}$, i.e., $\boldsymbol{\delta}_1^\top \otimes \mathbf{H}_k^{(1)} = \mathbf{L}\mathbf{J}_{\eta_{1,k}} + \boldsymbol{\delta}_k^\top \otimes \mathbf{G}^{(1)}$. Multiplying on the right by $\mathbf{G}^{-1}(\mathbf{A}_k')$ gives

$$\boldsymbol{\delta}_1^\top \otimes \big(\mathbf{H}_k\mathbf{G}^{-1}(\mathbf{A}_k')\big)^{(1)} = \mathbf{L}\mathbf{J}_{\eta_{1,k}}\mathbf{G}^{-1}(\mathbf{A}_k') + \boldsymbol{\delta}_k^\top \otimes \mathbf{A}_k'^{(1)},$$

and substituting the identity for $\mathbf{r}_k$ into the last term shows that $\mathsf{Aut}_k(\mathbf{c})$ is in simulation form with respect to $\mathbf{J}_{\eta_{1,k}}\mathbf{G}^{-1}(\mathbf{A}_k') + \widehat{\mathbf{R}}_k'$. □

The evaluation of F, including every lookup table, is a composition of the four primitive operations, and all multiplied payloads are monomials. Applying Lemma 10 along the evaluation of F therefore gives the following.

Corollary 1. *If all fresh encodings and RHS data are in simulation form, then $\mathbf{c}_F : [\![F(x^*)\mathbf{s}]\!]_{\mathbf{s}}^{\mathbf{A}_F}$ is in simulation form with respect to a matrix $\widehat{\mathbf{R}}_F$ computable from the programming randomness: $\boldsymbol{\delta}_1^\top \otimes \mathbf{A}_F^{(1)} = \mathbf{L}\widehat{\mathbf{R}}_F + F(x^*)\,\boldsymbol{\delta}_1^\top \otimes \mathbf{G}^{(1)}$.*

Bounding the simulation matrices. Condition on all Gaussian programming matrices $\mathbf{R}$ and $\mathbf{Z}_t$ having coefficient norms at most R_* and Z_*, respectively; Lemma 1 and a union bound over the polynomially many samples make failure negligible. The integral witnesses in hybrid H_2 are block embeddings of these matrices or unit selectors, giving bounds R_* for initial and digit encodings, Z_* for input gadget ciphertexts, and 1 for helpers and automorphism gadget ciphertexts.

A binary gadget product increases coefficient bounds by at most $D = 2\tau n$. For an input witness bounded by E, Lemma 10 gives

$$E_{\mathsf{Mul}} \le DE + 2\tau DR_* + Z_* :$$

each column of $\mathbf{S}$ sums 2τ terms bounded by DR_*, and the signed monomial μ preserves the norm of $\widehat{\mathbf{Z}}$. In Aut_k, automorphisms and block permutations preserve norms, the gadget witness is a unit selector, and the mask-switch term $\mathbf{J}_{\eta_{1,k}}\mathbf{G}^{-1}(\mathbf{A}_k')$ is bounded by D, so

$$E_{\mathsf{Aut}} \le DE + 2\tau DR_* + D + 1 \le D(E + 2\tau R_* + 2).$$

Since $D \ge 2$, at most b automorphisms give

$$D^b E + (2\tau R_* + 2)\sum_{j=1}^{b} D^j \le D^b(E + 4\tau R_* + 4).$$

Each lookup sums W such witnesses after scalar multiplication, whose binary gadget decomposition contributes at most D regardless of the table entries. Hence both lookup variants satisfy

$$E_{\mathsf{LUT}} \le W D^{b+1}(E + 4\tau R_* + 4).$$

One Mul and one lookup per branching-program step therefore yield

$$E_0 \le R_*, \qquad E_{t+1} \le \gamma E_t + \Delta_{\mathrm{sim}},$$

and, since $\gamma \ge 2$,

$$\|\widehat{\mathbf{R}}_F\|_\infty \le E_T \le \gamma^T R_* + \Delta_{\mathrm{sim}}\frac{\gamma^T - 1}{\gamma - 1} \le B_{\widehat{\mathbf{R}}}.$$

On the same tail event, this holds for all predicates of length at most T and lookup width W, including adaptive queries, without a further union bound over predicates.

Recall from key generation that, for the output matrix $\mathbf{A}_F = \mathsf{EvalPK}(F, \mathsf{mpk})$, the key matrix is

$$\mathbf{M}_F = \left[\ \mathbf{L}\ \middle|\ \boldsymbol{\delta}_1^\top \otimes \mathbf{A}_F^{(1)}\ \right] \in \mathcal{R}_q^{W\times N}, \qquad N = N_L + 2\tau,$$

and that a key is a sample from $D_{\Lambda_{\mathbf{y}}(\mathbf{M}_F),\sigma}$ for the target $\mathbf{y} = d_1\boldsymbol{\delta}_1$. For a key query with $F(x^*) = 1$, the gadget term of the corollary provides a gadget trapdoor for $\mathbf{M}_F$. For a matrix $\mathbf{M}$ over $\mathcal{R}$, its coefficient embedding replaces every entry by the $n \times n$ integer matrix of multiplication by it; we write $s_1(\mathbf{M})$ for the largest singular value of this embedding, i.e., its spectral norm, and $\|\mathbf{M}\|_F$ for its Frobenius norm, the square root of the sum of the squares of all its entries. Then $s_1(\mathbf{M}) \le \|\mathbf{M}\|_F$.

Lemma 11. *Let $F(x^*) = 1$, and let the entries of $\widehat{\mathbf{R}}_F$ have coefficients at most $B_{\widehat{\mathbf{R}}} \ge 1$. Then, given $\widehat{\mathbf{R}}_F$ and without a trapdoor of $\mathbf{B}^{(1)}$, one can compute a gadget trapdoor $\mathbf{X}$ of $\mathbf{M}_F$, i.e., a matrix with $\mathbf{M}_F\mathbf{X} = \mathbf{I}_W \otimes \mathbf{g}$, such that $s_1(\mathbf{X}) \le 3(2\tau n)^b N n B_{\widehat{\mathbf{R}}}$.*

Proof. For a matrix $\mathbf{Y}$ with one row and 2τ columns, let $\widetilde{\mathbf{G}}^{-1}(\mathbf{Y})$ be the binary matrix with $\mathbf{G}^{(1)}\widetilde{\mathbf{G}}^{-1}(\mathbf{Y}) = \mathbf{Y}$, i.e., $\mathbf{g}^{-1}(\mathbf{Y})$ padded with zero rows. We construct, for every $i \in \mathcal{H}$, a matrix $\mathbf{V}_i$ with $\mathbf{M}_F\mathbf{V}_i = \boldsymbol{\delta}_i^\top \otimes \mathbf{G}^{(1)}$, that is, a gadget in the row indexed by i.

The row indexed by 1. Let $\mathbf{V}_1 := \left(\begin{smallmatrix} -\widehat{\mathbf{R}}_F \\ \mathbf{I}_{2\tau} \end{smallmatrix}\right)$, where the upper block corresponds to the columns of $\mathbf{L}$ and the lower block to the last 2τ columns of $\mathbf{M}_F$. Then $\mathbf{M}_F\mathbf{V}_1 = -\mathbf{L}\widehat{\mathbf{R}}_F + \boldsymbol{\delta}_1^\top \otimes \mathbf{A}_F^{(1)}$, which equals $\boldsymbol{\delta}_1^\top \otimes \mathbf{G}^{(1)}$ by Corollary 1 and $F(x^*) = 1$.

Moving the gadget from row j to row $j \cdot a$. Let $j \in \mathcal{H}$ and $a \in \mathcal{K}$, and suppose that $\mathbf{M}_F\mathbf{V}_j = \boldsymbol{\delta}_j^\top \otimes \mathbf{G}^{(1)}$. Since $a \in \mathcal{J}$, $\mathbf{L}$ contains the block $\eta_{j,a}$. The matrix $\mathbf{J}_{\eta_{j,a}} \in \{0,1\}^{N_L\times 2\tau}$ selects this block of columns of $\mathbf{L}$, and we extend it to the N columns of $\mathbf{M}_F = [\,\mathbf{L} \mid \boldsymbol{\delta}_1^\top \otimes \mathbf{A}_F^{(1)}\,]$ by the block column vector

$$\widetilde{\mathbf{J}}_{\eta_{j,a}} := \begin{pmatrix} \mathbf{J}_{\eta_{j,a}} \\ \mathbf{0}_{2\tau\times 2\tau} \end{pmatrix} \in \{0,1\}^{N\times 2\tau},$$

whose upper block corresponds to the columns of $\mathbf{L}$ and whose lower block to the last 2τ columns of $\mathbf{M}_F$. Then, by block multiplication and the definition (3) of the block $\eta_{j,a}$ of $\mathbf{L}$,

$$\mathbf{M}_F\widetilde{\mathbf{J}}_{\eta_{j,a}} = \mathbf{L}\mathbf{J}_{\eta_{j,a}} = \boldsymbol{\delta}_j^\top \otimes \sigma_j(\mathbf{H}_a^{(1)}) - \boldsymbol{\delta}_{j\cdot a}^\top \otimes \mathbf{G}^{(1)}.$$

This block thus links the row indexed by j to the row indexed by $j \cdot a$. Let

$$\mathbf{V}_{j\cdot a} := \mathbf{V}_j\widetilde{\mathbf{G}}^{-1}\big(\sigma_j(\mathbf{H}_a^{(1)})\big) - \widetilde{\mathbf{J}}_{\eta_{j,a}}.$$

For the first term, $\mathbf{M}_F\mathbf{V}_j\widetilde{\mathbf{G}}^{-1}\big(\sigma_j(\mathbf{H}_a^{(1)})\big) = (\boldsymbol{\delta}_j^\top \otimes \mathbf{G}^{(1)})\widetilde{\mathbf{G}}^{-1}\big(\sigma_j(\mathbf{H}_a^{(1)})\big) = \boldsymbol{\delta}_j^\top \otimes \sigma_j(\mathbf{H}_a^{(1)})$ by the definition of $\widetilde{\mathbf{G}}^{-1}$. Subtracting the second term therefore cancels $\boldsymbol{\delta}_j^\top \otimes \sigma_j(\mathbf{H}_a^{(1)})$:

$$\mathbf{M}_F\mathbf{V}_{j\cdot a} = \boldsymbol{\delta}_j^\top \otimes \sigma_j(\mathbf{H}_a^{(1)}) - \big(\boldsymbol{\delta}_j^\top \otimes \sigma_j(\mathbf{H}_a^{(1)}) - \boldsymbol{\delta}_{j\cdot a}^\top \otimes \mathbf{G}^{(1)}\big) = \boldsymbol{\delta}_{j\cdot a}^\top \otimes \mathbf{G}^{(1)}.$$

All rows. Since $\mathcal{K} = \{g_0, \dots, g_{b-1}\}$ is a binary basis of $\mathcal{H}$, every $i \in \mathcal{H}$ is a product $i = a_1a_2\cdots a_t$ of $t \le b$ distinct elements $a_1, \dots, a_t \in \mathcal{K}$. Starting from $\mathbf{V}_1$ and applying the previous step with $(j,a) = (1,a_1), (a_1,a_2), (a_1a_2,a_3), \dots$, we obtain $\mathbf{V}_{a_1}, \mathbf{V}_{a_1a_2}, \dots, \mathbf{V}_i$ after $t \le b$ steps.

The matrix $\mathbf{X}$. Since $\mathbf{G}^{(1)} = (\mathbf{g}, \mathbf{0})$, the first τ columns of $\mathbf{V}_i$ satisfy $\mathbf{M}_F\mathbf{V}_i[\,\cdot, 1..\tau] = \boldsymbol{\delta}_i^\top \otimes \mathbf{g}$. Concatenating these columns for all $i \in \mathcal{H}$, in the order of the rows, gives $\mathbf{X}$ with $\mathbf{M}_F\mathbf{X} = \mathbf{I}_W \otimes \mathbf{g}$.

For the norm, the binary matrix $\widetilde{\mathbf{G}}^{-1}(\cdot)$ has τ nonzero rows of 2τ entries with binary coefficients; its spectral norm in the coefficient embedding is at most $\sqrt{2}\tau n \le 2\tau n$. Hence each step

gives $s_1(\mathbf{V}_{j\cdot a}) \le 2\tau n\, s_1(\mathbf{V}_j) + 1$, and with $s_1(\mathbf{V}_1) \le s_1(\widehat{\mathbf{R}}_F) + 1$ and $2\tau n \ge 2$, every $\mathbf{V}_i$ satisfies $s_1(\mathbf{V}_i) \le (2\tau n)^b (s_1(\widehat{\mathbf{R}}_F) + 2)$. Since $\mathbf{X}$ is the concatenation of W such blocks of columns, $s_1(\mathbf{X}) \le \sqrt{W}\max_i s_1(\mathbf{V}_i) \le \sqrt{W}(2\tau n)^b\big(s_1(\widehat{\mathbf{R}}_F) + 2\big)$. Finally, $\widehat{\mathbf{R}}_F$ has at most $N \cdot 2\tau$ entries, and the multiplication matrix of each entry consists of n^2 integers of absolute value at most $B_{\widehat{\mathbf{R}}}$; hence $s_1(\widehat{\mathbf{R}}_F) \le \|\widehat{\mathbf{R}}_F\|_F \le \sqrt{2\tau N}\, nB_{\widehat{\mathbf{R}}}$. Since $N \ge 2\tau W$, we have $\sqrt{W}\sqrt{2\tau N} \le N$, and hence $\sqrt{W}\, s_1(\widehat{\mathbf{R}}_F) \le NnB_{\widehat{\mathbf{R}}}$ and $2\sqrt{W} \le 2NnB_{\widehat{\mathbf{R}}}$, which give $s_1(\mathbf{X}) \le 3(2\tau n)^b NnB_{\widehat{\mathbf{R}}}$. □

By Lemma 11 and the key width condition above, $\sigma \ge s_1(\mathbf{X}) \cdot \omega(\sqrt{\log n})$, and $\mathsf{SamplePre}(\mathbf{M}_F, \mathbf{X}, \mathbf{y}, \sigma)$ therefore samples $D_{\Lambda_{\mathbf{y}}(\mathbf{M}_F),\sigma}$ up to negligible distance by Lemma 4. The resulting distribution is statistically close to the one produced by KeyGen.

We are now ready to define the hybrids.

Hybrid H_0 is the experiment $\mathsf{Exp}^{\mathrm{wAH}}_{\mathcal{A},\beta}$.

Hybrid H_1 *(programming).* H_1 is identical to H_0 except that the challenger, which knows x^*, samples $\bar{s}$ and the noise of the challenge ciphertext before publishing mpk, and replaces uniform matrices by programmed ones:

- it programs the rows $\mathbf{a}_t := \mathbf{B}^{(1)}\mathbf{Z}_t + X^{wx_t^*}\mathbf{G}^{(1)}$ of mpk for $\mathbf{Z}_t \leftarrow\!\$\ D_{\mathcal{R}^{m_B\times 2\tau},\sigma_Z}$; since $\mathbf{sG} = \bar{s}\mathbf{G}^{(1)} + \mathbf{G}^{(2)}$, the attribute ciphertexts then become $\mathbf{C}^{\mathrm{in}}_t = \left(\begin{smallmatrix}\mathbf{B}^{(1)}\mathbf{Z}_t \\ -\bar{s}\mathbf{B}^{(1)}\mathbf{Z}_t + \mathbf{e}_t\end{smallmatrix}\right) + X^{wx_t^*}\mathbf{G}$;
- it programs $\mathbf{A}_0 := \mathbf{BR}_0 + \mathbf{G}$, $\mathbf{A}^{\mathrm{in}}_{t,r,j,i} := \mathbf{BR} + d\mathbf{G}$ with the digits d of $\mathbf{C}^{\mathrm{in}}_t$, and $\mathbf{A}^{\mathrm{aut}}_{k,r,j,i} := \sigma_k(\mathbf{B})\mathbf{R} + d\mathbf{G}$ with the digits d of $\mathbf{C}_k$, for fresh $\mathbf{R} \leftarrow\!\$\ D_{\mathcal{R}^{m_B\times 2\tau},\sigma_{\mathbf{R}}}$.

The encodings are still computed as in Enc. Since $\mathbf{B}$ is close to uniform by Lemma 4, and since each $\mathbf{Z}_t$ and $\mathbf{R}$ is fresh and used nowhere else, Lemma 2 shows that $\mathbf{B}^{(1)}\mathbf{Z}_t$, and hence $\mathbf{a}_t$, and the programmed matrices are close to uniform. Thus $\mathsf{H}_0 \approx_s \mathsf{H}_1$.

Hybrid H_2 *(encodings from* $\mathbf{b}$*).* H_2 is identical to H_1 except that the challenger computes the components that involve the programmed matrices from $\mathbf{b}$:

- it sets $\mathbf{C}^{\mathrm{in}}_t := \mathbf{PZ}_t + \binom{\mathbf{0}}{\mathbf{e}_t} + X^{wx_t^*}\mathbf{G}$ with $\mathbf{P} := \mathbf{B} - \binom{\mathbf{0}}{\mathbf{b}}$, where $\mathbf{e}_t \leftarrow\!\$\ \chi_{\mathrm{fl}}^{1\times 2\tau}$ is the error of $\mathbf{C}^{\mathrm{in}}_t$ in H_1;
- it computes each programmed encoding under the mask $\mathbf{s}_i$ as $\sigma_i(\mathbf{b})\mathbf{R} + \mathbf{e}'$, where $\mathbf{e}' \leftarrow\!\$\ \chi_{\mathrm{fl}}^{1\times 2\tau}$ is the error of this encoding in H_1, i.e., in Enc.

Since $\mathbf{PZ}_t = \left(\begin{smallmatrix}\mathbf{B}^{(1)}\mathbf{Z}_t \\ -\bar{s}\mathbf{B}^{(1)}\mathbf{Z}_t - \mathbf{e}_b\mathbf{Z}_t\end{smallmatrix}\right)$ and $\sigma_i(\mathbf{b})\mathbf{R} = \mathbf{s}_i(\mathbf{A} - d\mathbf{G}) + \sigma_i(\mathbf{e}_b)\mathbf{R}$, these changes only add the terms $-\mathbf{e}_b\mathbf{Z}_t$ and $\sigma_i(\mathbf{e}_b)\mathbf{R}$ to the errors $\mathbf{e}_t$ and $\mathbf{e}'$, which Enc already draws from χ_{fl}. Except with negligible probability, $\|\mathbf{e}_b\mathbf{Z}_t\|_\infty \le B_{e\mathbf{Z}}$ and $\|\sigma_i(\mathbf{e}_b)\mathbf{R}\|_\infty \le B_{e\mathbf{R}}$ for the bounds $B_{e\mathbf{Z}}$ and $B_{e\mathbf{R}}$ defined above. Since $\sigma_{\mathrm{fl}} \ge \lambda^{\omega(1)} \cdot \max(B_{e\mathbf{Z}}, B_{e\mathbf{R}})$, Lemma 3 shows that these changes are statistical. Thus $\mathsf{H}_1 \approx_s \mathsf{H}_2$. All fresh components are now in simulation form:

- a helper $\mathbf{h}_k : [\![\mathbf{s}_k]\!]^{\mathbf{H}_k}_{\mathbf{s}}$ for $k \in \mathcal{J}$, with respect to $\mathbf{J}_{\eta_{1,k}}$;
- the simulated $\mathbf{C}^{\mathrm{in}}_t$, with $\nu = X^{wx_t}$, with respect to $\mathbf{J}_{\beta_1}\mathbf{Z}_t$, since $(\mathbf{C}^{\mathrm{in}}_t)^{(1)} = \mathbf{B}^{(1)}\mathbf{Z}_t + \nu\mathbf{G}^{(1)}$;
- $\mathbf{C}_k : \mathsf{GC}_{\mathbf{s}_k}(\mathbf{s})$ for $k \in \mathcal{K}$, with $\nu = 1$, with respect to $\mathbf{J}_{\eta_{k,k^{-1}}}$, since $\mathbf{C}^{(1)}_k = \sigma_k(\mathbf{H}^{(1)}_{k^{-1}})$ and $k^{-1} \in \mathcal{J}$;
- a programmed encoding $\mathbf{c} = \sigma_i(\mathbf{b})\mathbf{R} + \mathbf{e}' : [\![\mu\mathbf{s}_i]\!]^{\mathbf{A}}_{\mathbf{s}_i}$ with $\mathbf{A} = \sigma_i(\mathbf{B})\mathbf{R} + \mu\mathbf{G}$, with respect to $\mathbf{J}_{\beta_i}\mathbf{R}$.

Hybrid H_3 *(keys without trapdoor).* H_3 is identical to H_2 except that the challenger answers key queries without the trapdoor $\mathbf{T}$ of $\mathbf{B}^{(1)}$. Keys are sampled by $\mathsf{SamplePre}(\mathbf{M}_F, \mathbf{X}, \mathbf{y}, \sigma)$ (Lemma 4) with the gadget trapdoor $\mathbf{X}$ of Lemma 11, and $\mathbf{B}^{(1)}$ is sampled uniformly without a trapdoor. Since both key samplers output $D_{\Lambda_{\mathbf{y}}(\mathbf{M}_F),\sigma}$ up to negligible distance, $\mathsf{H}_2 \approx_s \mathsf{H}_3$.

Hybrid H_4 *(Ring-LWE samples).* H_4 is identical to H_3 except that $\mathbf{b}$, $\{\mathbf{h}_k\}_{k\in\mathcal{J}}$, and $c_\mathsf{m} - \lfloor q/2\rceil m^*$, the components of the challenge ciphertext that depend on $\bar{s}$ directly, are replaced by uniform values.

We show that a distinguisher $\mathcal{D}$ between H_3 and H_4 yields an adversary $\mathcal{B}$ against Assumption 1 with the same advantage, for the parameters (n, q, χ_s, χ) of Subsection 5.1 and $m = m_B + 1 + \tau|\mathcal{J}|$. Given a challenge $(\mathbf{a}, \mathbf{v}, \{\mathbf{a}_k, \mathbf{v}_k\}_{k\in\mathcal{J}})$, the adversary $\mathcal{B}$ splits $\mathbf{a}$ and $\mathbf{v}$ into blocks of m_B, 1, and τ entries for each $k \in \mathcal{J}$, samples $\mathbf{B}^{(2)}$, d_2, and the matrices $\mathbf{H}_k^{(2)}$ uniformly, and sets

- $\mathbf{B}^{(1)} := \mathbf{a}_{[m_B]}$ and $\mathbf{b} := \mathbf{v}_{[m_B]} + \mathbf{B}^{(2)}$;
- $d_1 := \mathbf{a}_{m_B+1}$ and $c_\mathsf{m} := \mathbf{v}_{m_B+1} + d_2 + \lfloor q/2\rceil m^*$;
- for every $k \in \mathcal{J}$, the first row of $\mathbf{H}_k$ to $(\mathbf{a}_k, \mathbf{a}'_k)$, where $\mathbf{a}'_k$ is the block of $\mathbf{a}$ for k, and $\mathbf{h}_k := (\mathbf{v}_k,\ \mathbf{v}'_k - \mathbf{g}) + \mathbf{H}_k^{(2)}$, where $\mathbf{v}'_k$ is the block of $\mathbf{v}$ for k.

In H_3, every other component of the view is computed from $\mathbf{b}$, the helpers, and public data: the programmed encodings from $\sigma_i(\mathbf{b})$, the matrices $\mathbf{C}_t^{\text{in}}$ from $\mathbf{b}$, the matrices $\mathbf{C}_k$ from the helpers, and the keys by $\mathsf{SamplePre}$ with the gadget trapdoor $\mathbf{X}$ of Lemma 11, which $\mathcal{B}$ computes from its own programming randomness. The adversary $\mathcal{B}$ computes them in the same way, runs $\mathcal{D}$ on the resulting view, and outputs its guess. If the challenge consists of Ring-LWE samples, then $\mathbf{b} = \mathbf{s}\mathbf{B} + \mathbf{e}_b$, $c_\mathsf{m} = \mathbf{s}\mathbf{d}^\top + e_\mathsf{m} + \lfloor q/2\rceil m^*$, and $\mathbf{h}_k = \mathbf{s}\mathbf{H}_k - \mathbf{s}_k\mathbf{G} + \mathbf{e}_k$ with $\mathbf{s} = (\bar{s}, 1)$, and the view is distributed as in H_3; if it is uniform, these values are uniform, and the view is distributed as in H_4. Hence $\mathsf{H}_3 \approx_c \mathsf{H}_4$.

Hybrid H_5 *(uniform gadget ciphertexts).* H_5 replaces the gadget ciphertexts $\mathbf{C}_t^{\text{in}}$ of the attribute symbols, including their first rows $\mathbf{a}_t$ in mpk, together with the programmed matrices and their encodings, which depend on the digits of $\mathbf{C}_t^{\text{in}}$ and $\mathbf{C}_k$, by independent uniform values. To this end, H_5 is identical to H_4 except that the challenger samples $\mathbf{B}^{(1)}$ by $\mathsf{TrapGen}$ again and answers key queries by $\mathsf{SampleLeftBlock}$ with its trapdoor $\mathbf{T}$, as in KeyGen, and that these components are replaced by independent uniform values. First, since $\mathbf{b}$ is uniform in H_4, it no longer depends on $\mathbf{B}^{(1)}$, and $\mathbf{B}^{(1)}$ can again be sampled with a trapdoor: $\mathbf{B}^{(1)}$ from $\mathsf{TrapGen}$ is statistically close to uniform by Lemma 4, and both key samplers output $D_{\Lambda_\mathbf{y}(\mathbf{M}_F),\sigma}$ up to negligible distance by Lemmas 4 and 6; after this change, the keys no longer depend on the programming randomness $\mathbf{R}$ and $\mathbf{Z}_t$. For the same reason, the matrix $\binom{\mathbf{B}}{\mathbf{b}}$ is statistically close to uniform. By Lemma 2, which applies to matrices with three rows by the choice of m_B, $\sigma_\mathbf{R}$, and σ_Z in Subsection 5.1, each pair consisting of a programmed matrix $\sigma_i(\mathbf{B})\mathbf{R} + d\mathbf{G}$ and its encoding $\sigma_i(\mathbf{b})\mathbf{R} + \mathbf{e}'$ is statistically close to uniform, independently of d, since each uses a fresh $\mathbf{R}$; similarly, $\mathbf{P} = \mathbf{B} - \binom{\mathbf{0}}{\mathbf{b}}$ is statistically close to uniform, and hence $\mathbf{C}_t^{\text{in}} = \mathbf{P}\mathbf{Z}_t + \binom{\mathbf{0}}{\mathbf{e}_t} + X^{wx_t^*}\mathbf{G}$ is statistically close to uniform; the matrices $\mathbf{C}_k$ are determined by uniform helpers; and c_m is uniform. In H_5, the public key, the ciphertext, and the keys are therefore independent of (x^*, m^*).

Combining the steps, $\mathsf{H}_0 \approx_s \mathsf{H}_1 \approx_s \mathsf{H}_2 \approx_s \mathsf{H}_3 \approx_c \mathsf{H}_4 \approx_s \mathsf{H}_5$, and H_5 is identical for $\beta = 0$ and $\beta = 1$. Hence $\mathsf{Exp}^{\text{wAH}}_{\mathcal{A},0}$ and $\mathsf{Exp}^{\text{wAH}}_{\mathcal{A},1}$ are computationally indistinguishable, which proves Theorem 2. □

B Compressed Digit Encodings

This appendix gives the variant of the scheme of Subsection 5.1 with compressed digit encodings, outlined in Subsection 6.1, and proves its correctness and security. We describe only the differences from Section 5 and Appendix A; everything not mentioned is unchanged.

B.1 Decompression

Let $p = \prod_{i=1}^{L_p} p_i$ be a product of distinct primes $p_i \equiv 1 \pmod{2n}$, coprime to q, and let $Q := pq$. Elements of $\mathcal{R}_Q := \mathcal{R}/Q\mathcal{R}$ are represented as those of $\mathcal{R}_q$ in Section 3. Both moduli use the same ring $\mathcal{R}$, so that n is chosen for the larger modulus Q. We round coefficients to the nearest integer with ties away from zero, so that $\lfloor -x \rceil = -\lfloor x \rceil$. Since p is odd, every value rounded below is an integer divided by p and hence never a tie, so that $\lfloor K + y \rceil = K + \lfloor y \rceil$ for every integer K and such y. Recall the gadget matrix $\mathbf{G}_h = \mathbf{I}_h \otimes \mathbf{g}$ of Section 3, whose entries are integers smaller than q; in particular, $\mathbf{G} = \mathbf{G}_2$.

Definition 4 (Decompression)**.** For a matrix $\mathbf{M}$ over $\mathcal{R}_Q$ with h columns, let

$$\mathsf{Dc}(\mathbf{M}) := \lfloor \mathbf{M}\mathbf{G}_h/p \rceil \bmod q,$$

where $\mathbf{M}\mathbf{G}_h$ is computed in $\mathcal{R}_Q$, its representative is divided by p over the rationals, and the result is rounded coefficient-wise.

Since changing a representative by a multiple of Q changes the quotient by a multiple of q, Dc is well defined. Since σ_k permutes coefficients up to sign, which preserves these representatives for odd Q, and fixes $\mathbf{G}_h$, it commutes with Dc: $\sigma_k(\mathsf{Dc}(\mathbf{M})) = \mathsf{Dc}(\sigma_k(\mathbf{M}))$. Each entry of $\mathbf{M}$ is expanded into τ entries, one for each product with a power 2^u, before the division by p. We index the columns of $\mathsf{Dc}(\mathbf{M})$ by pairs $(l, u) \in [h] \times [0, \tau - 1]$, where l is the column of $\mathbf{M}$ and u the exponent of 2^u, which starts from 0 like the index i of the digits $d_{r,j,i}$ in Section 4; the pair (l, u) is the $((l-1)\tau + u + 1)$-th column. In particular, the columns of $\mathbf{G} = \mathbf{G}_2$ are indexed by $(c, u) \in [2] \times [0, \tau - 1]$. The columns with $u = 0$ form $\lfloor \mathbf{M}/p \rceil \bmod q$.

Let B_s be a bound with $\|\bar{s}\|_\infty \le B_s$ for $\bar{s} \leftarrow\!\$\ \chi_s$, except with negligible probability.

Lemma 12 (Decompression of encodings)**.** *Let* $\mathbf{s}' = (\bar{s}', 1)$ *with* $\|\bar{s}'\|_\infty \le B_s$*, let* $\mathbf{M} \in \mathcal{R}_Q^{2\times h}$ *and* $\mathbf{X} \in \mathcal{R}^{2\times h}$*, and let* $\check{\mathbf{c}} = \mathbf{s}'(\mathbf{M} - p\mathbf{X}) + \mathbf{e} \bmod Q$*. Then*

$$\mathsf{Dc}(\check{\mathbf{c}}) = \mathbf{s}'\big(\mathsf{Dc}(\mathbf{M}) - \mathbf{X}\mathbf{G}_h\big) + \mathbf{e}' \bmod q,$$

where $\|\mathbf{e}'\|_\infty \le (q/p)\|\mathbf{e}\|_\infty + nB_s/2 + 1$*.*

Proof. For $z \in \mathcal{R}_Q$, the quotient z/p of a representative is well defined modulo q, since $Q/p = q$; we compute modulo q with such quotients. Let $\mathbf{\Phi} := \mathbf{M}\mathbf{G}_h/p - \mathsf{Dc}(\mathbf{M})$, whose coefficients lie in $[-1/2, 1/2]$ by the definition of Dc. Dividing $\check{\mathbf{c}}\mathbf{G}_h \equiv \mathbf{s}'\mathbf{M}\mathbf{G}_h - p\mathbf{s}'\mathbf{X}\mathbf{G}_h + \mathbf{e}\mathbf{G}_h \pmod{Q}$ by p gives

$$\check{\mathbf{c}}\mathbf{G}_h/p \equiv \mathbf{s}'\mathsf{Dc}(\mathbf{M}) - \mathbf{s}'\mathbf{X}\mathbf{G}_h + \big(\mathbf{s}'\mathbf{\Phi} + \mathbf{e}\mathbf{G}_h/p\big) \pmod{q},$$

whose first two terms are integral. Rounding therefore gives $\mathsf{Dc}(\check{\mathbf{c}}) = \mathbf{s}'(\mathsf{Dc}(\mathbf{M}) - \mathbf{X}\mathbf{G}_h) + \mathbf{e}' \bmod q$ with $\mathbf{e}' := \lfloor \mathbf{s}'\mathbf{\Phi} + \mathbf{e}\mathbf{G}_h/p \rceil$. The value rounded to $\mathbf{e}'$ has three parts: $\bar{s}'\mathbf{\Phi}^{(1)}$, whose coefficients are sums of n products of a coefficient of $\bar{s}'$ and one of $\mathbf{\Phi}^{(1)}$ and hence at most $nB_s/2$; $\mathbf{\Phi}^{(2)}$, which the second entry 1 of $\mathbf{s}'$ leaves unchanged and whose coefficients are at most $1/2$; and $\mathbf{e}\mathbf{G}_h/p$, every entry of which is an entry of $\mathbf{e}$ times some $2^u < q$ divided by p and hence at most $(q/p)\|\mathbf{e}\|_\infty$. Rounding adds at most $1/2$, and therefore $\|\mathbf{e}'\|_\infty \le nB_s/2 + 1/2 + (q/p)\|\mathbf{e}\|_\infty + 1/2$. □

With $\mathbf{X} = d\,\mathbf{I}_2$ for a digit d, $\mathbf{X}\mathbf{G} = d\mathbf{G}$, and $\mathsf{Dc}(\check{\mathbf{c}})$ is an encoding $:[\![d\mathbf{s}']\!]_{\mathbf{s}'}^{\mathsf{Dc}(\mathbf{M})}$ in the sense of Section 4.

B.2 Construction

The scheme differs from that of Subsection 5.1 as follows. In the anchor, we follow [BTVW17] and define $(\mathbf{b}, \mathbf{B})$ modulo Q, so that rounding recovers the anchor modulo q. Since p and q are odd, every $z \in (-Q/2, Q/2]$ is uniquely $z = pz' + r$ with $z' \in (-q/2, q/2]$ and $r \in (-p/2, p/2]$, and then $z' = \lfloor z/p \rceil$.

Setup. Setup changes the following components of mpk.

- *Anchor:* it samples $\mathbf{B}$ as in Subsection 5.1, samples $\mathbf{r} \in \mathcal{R}^{2\times m_B}$ with coefficients uniform in $(-p/2, p/2]$, and publishes $\mathbf{B}_Q := p\mathbf{B} + \mathbf{r} \bmod Q$ instead of $\mathbf{B}$, where $\mathbf{B}$ is taken with its representative. Then $\lfloor \mathbf{B}_Q/p \rceil \bmod q = \mathbf{B}$, and $\mathbf{B}_Q$ is within the same statistical distance from uniform over $\mathcal{R}_Q^{2\times m_B}$ as $\mathbf{B}$ from uniform over $\mathcal{R}_q^{2\times m_B}$.
- *Compressed digit matrices:* it samples $\check{\mathbf{A}}^{\mathrm{in}}_{t,r,j,i} \leftarrow\!\$\, \mathcal{R}_Q^{2\times 2}$ and $\check{\mathbf{A}}^{\mathrm{aut}}_{k,r,j,i} \leftarrow\!\$\, \mathcal{R}_Q^{2\times 2}$ instead of $\mathbf{A}^{\mathrm{in}}_{t,r,j,i}$ and $\mathbf{A}^{\mathrm{aut}}_{k,r,j,i}$.

The digit matrices of Section 4 are $\mathbf{A}^{\mathrm{in}}_{t,r,j,i} := \mathsf{Dc}(\check{\mathbf{A}}^{\mathrm{in}}_{t,r,j,i})$ and $\mathbf{A}^{\mathrm{aut}}_{k,r,j,i} := \mathsf{Dc}(\check{\mathbf{A}}^{\mathrm{aut}}_{k,r,j,i})$. All of these matrices are computed from mpk.

Encryption. Enc samples $\mathbf{e}_Q \leftarrow\!\$\, \chi^{1\times m_B}$ and computes the anchor

$$\mathbf{b}_Q := \mathbf{s}\mathbf{B}_Q + \mathbf{e}_Q \bmod Q$$

instead of $\mathbf{b}$, from which anyone computes $\mathbf{b} := \lfloor \mathbf{b}_Q/p \rceil \bmod q$. Since $\mathbf{b}$ consists of the columns with $u = 0$ of $\mathsf{Dc}(\mathbf{b}_Q)$, Lemma 12 with $\mathbf{X} = \mathbf{0}$ gives $\mathbf{b} = \mathbf{s}\mathbf{B} + \mathbf{e}_b$ with $\|\mathbf{e}_b\|_\infty \le (q/p)B_\chi + nB_s/2 + 1$. For every digit d of Subsection 5.1, with mask $\mathbf{s}' \in \{\mathbf{s}, \mathbf{s}_k\}$ and compressed matrix $\check{\mathbf{A}}$, it samples $\check{\mathbf{e}} \leftarrow\!\$\, \chi_{\mathrm{fl}}^{1\times 2}$ and computes the compressed digit encoding

$$\check{\mathbf{c}} := \mathbf{s}'\big(\check{\mathbf{A}} - p\,d\,\mathbf{I}_2\big) + \check{\mathbf{e}} \bmod Q$$

instead of the digit encoding of Subsection 5.1. The ciphertext contains $\mathbf{b}_Q$ and the compressed digit encodings instead of $\mathbf{b}$ and the digit encodings.

Evaluation. The evaluator decompresses every compressed digit encoding into $\mathsf{Dc}(\check{\mathbf{c}}) : [\![d\mathbf{s}']\!]_{\mathbf{s}'}^{\mathsf{Dc}(\check{\mathbf{A}})}$ by Lemma 12, and then runs EvalCT as in Section 4.

Key generation. Key generation also uses the anchor after decompression. For a parameter $m_c \le m_B$, let $\mathbf{B}_{Q,[m_c]} \in \mathcal{R}_Q^{2\times m_c}$ consist of the first m_c columns of $\mathbf{B}_Q$, and $\mathbf{b}_{Q,[m_c]}$ and $\mathbf{e}_{Q,[m_c]}$ of the first m_c entries of $\mathbf{b}_Q$ and $\mathbf{e}_Q$, and let $\mathbf{B}^{\mathrm{dc}} := \mathsf{Dc}(\mathbf{B}_{Q,[m_c]}) \in \mathcal{R}_q^{2\times m_c\tau}$. Key generation replaces $\mathbf{L}$ of (3) by

$$\widetilde{\mathbf{L}} := \Big[\ \mathbf{L} \ \Big|\ \big(\boldsymbol{\delta}_i^\top \otimes \sigma_i(\mathbf{B}^{\mathrm{dc}(1)})\big)_{i\in\mathcal{H}} \ \Big|\ \mathbf{I}_W\ \Big] \in \mathcal{R}_q^{W\times \widetilde{N}_L}, \qquad \widetilde{N}_L := N_L + Wm_c\tau + W,$$

and accordingly $\mathbf{M}_F$ by $\widetilde{\mathbf{M}}_F := [\,\widetilde{\mathbf{L}} \mid \boldsymbol{\delta}_1^\top \otimes \mathbf{A}_F^{(1)}\,]$ with $\widetilde{N} := \widetilde{N}_L + 2\tau$ columns. We label the new column blocks β_i^{dc} for $\boldsymbol{\delta}_i^\top \otimes \sigma_i(\mathbf{B}^{\mathrm{dc}(1)})$ and id for $\mathbf{I}_W$; Subsection B.4 explains why the security proof needs them. The blocks $\boldsymbol{\delta}_i^\top \otimes \sigma_i(\mathbf{B}^{(1)})$ still form the block-diagonal part of $\mathsf{SampleLeftBlock}$, which samples $\mathbf{k}_F$ with the trapdoor $\mathbf{T}$ of $\mathbf{B}^{(1)} = \lfloor \mathbf{B}_Q^{(1)}/p \rceil$ as before.

Decryption. Decryption computes $\mathbf{b} = \lfloor \mathbf{b}_Q/p \rceil$ and $\mathbf{b}^{\mathrm{dc}} := \mathsf{Dc}(\mathbf{b}_{Q,[m_c]})$, and replaces the rows of Subsection 5.1 by

$$\widetilde{\mathbf{y}}_F := \Big(\big(\sigma_i(\mathbf{b})\big)_i,\ \big(\sigma_i(\mathbf{h}_k)\big)_{i,k},\ \big(\sigma_i(\mathbf{b}^{\mathrm{dc}})\big)_i,\ \mathbf{0}_W,\ \mathbf{c}_F \Big),$$
$$\widetilde{\boldsymbol{\kappa}}_F := \Big(\big(\sigma_i(\mathbf{B}^{(2)})\big)_i,\ \big(\sigma_i(\mathbf{H}_k^{(2)}) - \mathbf{G}^{(2)}\big)_{i,k},\ \big(\sigma_i(\mathbf{B}^{\mathrm{dc}(2)})\big)_i,\ \mathbf{0}_W,\ \mathbf{A}_F^{(2)} \Big),$$

with $i \in \mathcal{H}$ and $k \in \mathcal{J}$, whose entries match the columns of $\widetilde{\mathbf{M}}_F$. The rest is unchanged.

Parameters. The parameter conditions in Appendix A change as follows, where $\varrho_Q := (L_q + L_p)\log(2n) + 2\lambda$ and $\sigma_{\mathbf{R}}$ lies in the range of Lemma 2 for the modulus Q. Let $B'_\chi := \frac{q}{p} B_\chi + \frac{nB_s}{2} + 1$, which bounds the errors of $\mathbf{b}$ and $\mathbf{b}^{\mathrm{dc}}$ by Lemma 12.

1. *Hardness:* circular Ring-LWE (Assumption 1) is hard for (n, Q, χ_s, χ), with the same gadget vector $\mathbf{g}$.
2. *Uniformity:* additionally, $m_c(\log \sigma_{\mathbf{R}} - 1) \geq 3\log Q + \varrho_Q$.
3. *Flooding:* B_χ is replaced by B'_χ in $B_{e\mathbf{R}}$ and $B_{e\mathbf{Z}}$, and the maximum additionally includes $B_{e\check{\mathbf{R}}} := \sqrt{m_c n}\, B_\chi \sigma_{\mathbf{R}} \sqrt{\lambda}$.
4. *Key width:* N is replaced by $\widetilde{N}$, and $B_{\widehat{\mathbf{R}}}$ is given by (5) with every occurrence of R_* replaced by
$$R_*^{\mathrm{dc}} := \max\left\{ R_*, \frac{m_c n R_* + 1}{2} \right\}.$$
Here D, Z_*, and γ are unchanged; the replacement accounts for the witnesses of decompressed digit encodings in Lemma 13.
5. *Correctness:* $B_\chi + \widetilde{N} n \sigma \sqrt{\lambda} \cdot \max(B'_\chi, B_F, B_s) < \lfloor q/4 \rceil$, where B_F is taken with $\beta_{\mathsf{RHS}} = \frac{q}{p}\sigma_{\mathrm{fl}}\sqrt{\lambda} + \frac{nB_s}{2} + 1$.

Efficiency. A compressed digit encoding consists of 2 elements of $\mathcal{R}_Q$ instead of 2τ elements of $\mathcal{R}_q$. The RHS data of a gadget ciphertext therefore consist of $8\tau^2$ elements of $\mathcal{R}_Q$ instead of $8\tau^3$ elements of $\mathcal{R}_q$, a reduction by a factor of about $\tau/2$ in bits for the same ring dimension. The anchor $\mathbf{b}_Q$ consists of m_B elements of $\mathcal{R}_Q$ instead of $\mathcal{R}_q$, and the key additionally contains $W(m_c\tau + 1)$ short ring elements.

B.3 Correctness

In (4), the rows $\widetilde{\mathbf{y}}_F$ and $\widetilde{\boldsymbol{\kappa}}_F$ and the matrix $\widetilde{\mathbf{M}}_F$ satisfy the same identity, with an extended error row $\widetilde{\mathbf{e}}_F$. The entries $\sigma_i(\mathbf{b})$ now have errors at most B'_χ. For the new entries, Lemma 12 with $\mathbf{X} = \mathbf{0}$ gives $\sigma_i(\mathbf{b}^{\mathrm{dc}}) - \sigma_i(\mathbf{B}^{\mathrm{dc}(2)}) = \sigma_i(\bar{s})\sigma_i(\mathbf{B}^{\mathrm{dc}(1)}) + \sigma_i(\mathbf{e}^{\mathrm{dc}})$ with $\|\mathbf{e}^{\mathrm{dc}}\|_\infty \leq B'_\chi$, and the zero entries satisfy $\mathbf{0}_W = \bar{\mathbf{s}}_{\mathcal{H}}\mathbf{I}_W - \bar{\mathbf{s}}_{\mathcal{H}}$, whose error $-\bar{\mathbf{s}}_{\mathcal{H}}$ has norm at most B_s. By Lemma 12, the decompressed digit encodings have errors at most β_{RHS} as in the correctness condition, and Lemma 9 applies to them. The proof of Theorem 1 then carries over with $\|\widetilde{\mathbf{e}}_F\|_\infty \leq \max(B'_\chi, B_F, B_s)$ and $\widetilde{N}$ in place of N.

B.4 Security

Theorem 3. *Under circular Ring-LWE with automorphism keys (Assumption 1) for (n, Q, χ_s, χ), the scheme of this appendix is selectively weakly attribute-hiding (Definition 1).*

The proof follows Appendix A with $\widetilde{\mathbf{L}}$ in place of $\mathbf{L}$. We first extend the tools and then describe the changes in the hybrids.

Simulation forms. Simulation forms are defined as in Appendix A with respect to $\widetilde{\mathbf{L}}$. As in Appendix A, $\mathbf{J}_\theta$ denotes the 0/1 matrix that embeds a block into the columns of a column block θ of $\widetilde{\mathbf{L}}$. For the new blocks, $\mathbf{J}_{\beta_i^{\mathrm{dc}}} \in \{0,1\}^{\widetilde{N}_L \times m_c\tau}$ and $\mathbf{J}_{\mathrm{id}} \in \{0,1\}^{\widetilde{N}_L \times W}$ select the columns of β_i^{dc} and id, so that $\widetilde{\mathbf{L}}\mathbf{J}_{\beta_i^{\mathrm{dc}}} = \boldsymbol{\delta}_i^\top \otimes \sigma_i(\mathbf{B}^{\mathrm{dc}(1)})$ and $\widetilde{\mathbf{L}}\mathbf{J}_{\mathrm{id}} = \mathbf{I}_W$; in particular, $\widetilde{\mathbf{L}}\mathbf{J}_{\mathrm{id}}\mathbf{Y} = \mathbf{Y}$ for every $\mathbf{Y}$ with W rows. Lemma 10, Corollary 1, and Lemma 11 hold with $\widetilde{\mathbf{L}}$, $\widetilde{\mathbf{M}}_F$, and $\widetilde{N}$ in place of $\mathbf{L}$, $\mathbf{M}_F$, and N, with the same proofs. The only step that refers to the blocks of $\mathbf{L}$ is the identity $\mathbf{P}_k\sigma_k(\mathbf{L}) = \mathbf{L}\boldsymbol{\Pi}_k$ in the case of Aut_k; it extends to $\widetilde{\mathbf{L}}$, since $\mathbf{P}_k\sigma_k$ maps the block β_j^{dc} to $\beta_{k\cdot j}^{\mathrm{dc}}$ by the same computation as for β_j, and maps $\mathbf{I}_W$ to $\mathbf{P}_k = \mathbf{I}_W\mathbf{P}_k$, a permutation of the columns of id. The proof of Lemma 11 uses only the blocks $\eta_{j,a}$ and $\widehat{\mathbf{R}}_F$, and its norm bound uses only $\widetilde{N} \geq 2\tau W$.

The following lemma replaces the simulation form of the programmed digit encodings: a compressed digit matrix programmed with a short $\mathbf{R}$ decompresses into a short combination of the columns of $\mathbf{B}^{\mathrm{dc}}$ plus a small rounding error, which the block $\mathbf{I}_W$ absorbs.

Lemma 13 (Decompressed programmed matrices)**.** *Let $i \in \mathcal{H}$, $d \in \mathcal{R}$, and $\mathbf{R} \in \mathcal{R}^{m_c\times 2}$ with $\|\mathbf{R}\|_\infty \leq B_{\mathbf{R}}$, and let $\check{\mathbf{A}} := \sigma_i(\mathbf{B}_{Q,[m_c]})\mathbf{R} + p\,d\,\mathbf{I}_2 \bmod Q$ be the compressed digit matrix programmed with them. Then*

$$\boldsymbol{\delta}_i^\top \otimes \mathsf{Dc}(\check{\mathbf{A}})^{(1)} - \boldsymbol{\delta}_i^\top \otimes d\mathbf{G}^{(1)} = \widetilde{\mathbf{L}}\big(\mathbf{J}_{\beta_i^{\mathrm{dc}}}\mathbf{S}_{\mathbf{R}} + \mathbf{J}_{\mathrm{id}}(\boldsymbol{\delta}_i^\top \otimes \mathbf{E})\big),$$

where $\mathbf{E} \in \mathcal{R}^{1\times 2\tau}$ satisfies $\|\mathbf{E}\|_\infty \leq (m_c n B_{\mathbf{R}} + 1)/2$, and $\mathbf{S}_{\mathbf{R}} \in \mathcal{R}^{m_c\tau\times 2\tau}$ has the entry $R_{l,c}$ in the row (l,u) and the column (c,u) for $l \in [m_c]$, $c \in [2]$, and $u < \tau$, and zeros elsewhere.

Proof. As in the proof of Lemma 12, we compute modulo q with quotients by p; rounding such a quotient is well defined modulo q. Since $p\,d\,\mathbf{I}_2\mathbf{G}/p = d\mathbf{G}$ is integral, $\mathsf{Dc}(\check{\mathbf{A}}) = \lfloor \sigma_i(\mathbf{B}_{Q,[m_c]})\mathbf{R}\mathbf{G}/p \rceil + d\mathbf{G} \bmod q$. The entry of $\lfloor \sigma_i(\mathbf{B}_{Q,[m_c]})\mathbf{R}\mathbf{G}/p \rceil$ in the first row and the column (c,u) is $\lfloor \sum_l V_{l,u}R_{l,c} \rceil$ with $V_{l,u} := \sigma_i(\mathbf{B}_{Q,[m_c]})_l^{(1)} 2^u/p$, where $\sigma_i(\mathbf{B}_{Q,[m_c]})_l^{(1)}$ is the l-th entry of the first row. By the definition of Dc and its commutation with σ_i, $\lfloor V_{l,u} \rceil \equiv \sigma_i(\mathbf{B}^{\mathrm{dc}(1)})_{(l,u)} \pmod q$. Writing $V_{l,u} = \lfloor V_{l,u} \rceil + \Phi_{l,u}$ with $\|\Phi_{l,u}\|_\infty \leq 1/2$, the sum $\sum_l \lfloor V_{l,u} \rceil R_{l,c}$ is integral, and hence

$$\lfloor \textstyle\sum_l V_{l,u}R_{l,c} \rceil = \sum_l \sigma_i(\mathbf{B}^{\mathrm{dc}(1)})_{(l,u)}R_{l,c} + E_{(c,u)} \pmod q, \qquad E_{(c,u)} := \lfloor \textstyle\sum_l \Phi_{l,u}R_{l,c} \rceil.$$

The first sum is the column (c,u) of $\sigma_i(\mathbf{B}^{\mathrm{dc}(1)})\mathbf{S}_{\mathbf{R}}$, and $\|E_{(c,u)}\|_\infty \leq m_c n B_{\mathbf{R}}/2 + 1/2$. Placing both sides in the row indexed by i gives the claim. □

Coefficient bound for key simulation. On the Gaussian tail event of Appendix A, Lemma 13 applies with $B_{\mathbf{R}} = R_*$. The blocks selected by $\mathbf{J}_{\beta_i^{\mathrm{dc}}}$ and $\mathbf{J}_{\mathrm{id}}$ have disjoint rows in the witness, so the decompressed digit witness has coefficient norm at most $\max\{R_*, (m_c n R_* + 1)/2\} = R_*^{\mathrm{dc}}$. The uncompressed initial encoding still has bound $R_* \leq R_*^{\mathrm{dc}}$, while input gadget ciphertexts and unit-selector witnesses retain bounds Z_* and 1. The recurrence following Corollary 1 thus applies with R_*^{dc} in place of R_*, both in the initial bound and in the digit-witness terms of every Mul and Aut_k. Consequently, the modified $B_{\widehat{\mathbf{R}}}$ bounds the output witness and suffices for Lemma 11 with $\widetilde{N}$.

The block β_i^{dc} is needed because the columns of $\mathbf{B}^{\mathrm{dc}}$ with $u \geq 1$ are not short combinations of those of $\mathbf{B}$: decompression multiplies by $\mathbf{G}$ before the division by p, which amplifies the fraction $\mathbf{r}/p$. The block $\mathbf{I}_W$ absorbs $\mathbf{E}$, as in the trapdoor extension in Hermite normal form of [BTVW17], so that Lemma 11 still yields an exact gadget trapdoor. The lemma holds for any gadget whose entries are smaller than q, including the RNS gadget of Subsection 6.1.

Hybrids. The hybrids are those of Appendix A with the following changes.

- H_1 *(programming).* The challenger also programs every compressed digit matrix as $\check{\mathbf{A}} := \sigma_i(\mathbf{B}_{Q,[m_c]})\mathbf{R} + p\,d\,\mathbf{I}_2 \bmod Q$ for the digit d of $\mathbf{C}_t^{\text{in}}$ or $\mathbf{C}_k$ and the mask $\mathbf{s}_i$ of the encoding, with a fresh $\mathbf{R} \leftarrow\!\$\; D_{\mathcal{R}^{m_c\times 2},\sigma_{\mathbf{R}}}$. Since $\mathbf{B}_Q$ is close to uniform, Lemma 2, applied over $\mathcal{R}_Q$ with the uniformity condition on m_c, shows that these matrices are close to uniform.
- H_2 *(encodings from* $\mathbf{b}$*).* The challenger computes every compressed digit encoding as $\check{\mathbf{c}} := \sigma_i(\mathbf{b}_{Q,[m_c]})\mathbf{R} + \check{\mathbf{e}} \bmod Q$. Since $\sigma_i(\mathbf{b}_{Q,[m_c]})\mathbf{R} = \mathbf{s}_i(\check{\mathbf{A}} - p\,d\,\mathbf{I}_2) + \sigma_i(\mathbf{e}_{Q,[m_c]})\mathbf{R}$, this adds a term of norm at most $B_{e\check{\mathbf{R}}}$ to the error $\check{\mathbf{e}}$, which Lemma 3 hides. The other components are computed from $\mathbf{b} = \lfloor \mathbf{b}_Q/p \rceil$ as in Appendix A, with the error $\mathbf{e}_b$ of norm at most B'_χ. The decompressed encodings are computed from $\check{\mathbf{c}}$ as in the scheme, and by Lemma 13 they are in simulation form with respect to $\mathbf{J}_{\beta_i^{\text{dc}}}\mathbf{S}_{\mathbf{R}} + \mathbf{J}_{\text{id}}(\boldsymbol{\delta}_i^\top \otimes \mathbf{E})$; these replace the programmed encodings in the list of simulation forms at the end of H_2.
- H_3 *(keys without trapdoor).* Unchanged, with $\widetilde{\mathbf{M}}_F$ and the gadget trapdoor of Lemma 11 for $\widetilde{\mathbf{L}}$; the matrix $\mathbf{B}_Q^{(1)}$ is sampled uniformly, and hence so is $\mathbf{B}^{(1)} = \lfloor \mathbf{B}_Q^{(1)}/p \rceil$.
- H_4 *(Ring-LWE samples).* H_4 replaces $\mathbf{b}_Q$ instead of $\mathbf{b}$ by a uniform value. The adversary $\mathcal{B}$ receives a challenge of Assumption 1 modulo Q, sets $\mathbf{B}_Q^{(1)}$ and $\mathbf{b}_Q$ from its first m_B entries as $\mathbf{B}^{(1)}$ and $\mathbf{b}$ in Appendix A, with $\mathbf{B}_Q^{(2)}$ uniform over $\mathcal{R}_Q$, and computes $\mathbf{B}$ and $\mathbf{b}$ by rounding. It reduces all other entries modulo q before using them as in Appendix A. Reduction modulo q maps a Ring-LWE sample $(\mathbf{a}, \bar{s}\mathbf{a} + \mathbf{e})$ modulo Q to one modulo q with the same secret and error, maps $\bar{s}\mathbf{a}_k - \sigma_k(\bar{s})\mathbf{g} + \mathbf{e}_k$ to the same expression modulo q, and maps uniform values to uniform values. The view is therefore distributed as in H_3 or as in H_4.
- H_5 *(uniform gadget ciphertexts).* The challenger samples $\mathbf{B}^{(1)}$ by $\mathsf{TrapGen}$ again and sets $\mathbf{B}_Q^{(1)} := p\mathbf{B}^{(1)} + \mathbf{r}^{(1)}$ as in Setup, which is statistically close by Lemma 4 and the decomposition $z = pz' + r$. Since $\mathbf{b}_Q$ is uniform, $\mathbf{b} = \lfloor \mathbf{b}_Q/p \rceil$ is uniform, and the argument of Appendix A applies to $\binom{\mathbf{B}}{\mathbf{b}}$. In addition, $\binom{\mathbf{B}_{Q,[m_c]}}{\mathbf{b}_{Q,[m_c]}}$ is close to uniform, and by Lemma 2 over $\mathcal{R}_Q$ with three rows, each pair of a compressed digit matrix $\sigma_i(\mathbf{B}_{Q,[m_c]})\mathbf{R} + p\,d\,\mathbf{I}_2$ and its encoding $\sigma_i(\mathbf{b}_{Q,[m_c]})\mathbf{R} + \check{\mathbf{e}}$ is close to uniform, independently of d, since each uses a fresh $\mathbf{R}$. These pairs replace the programmed pairs in the argument of Appendix A; the decompressed matrices and encodings are functions of them and need no separate treatment.

The remaining steps are as in Appendix A, which proves Theorem 3. □