

Trace-Factored BigSwitch for Matrix-Friendly FHE

Dong Jin Park, Hyunseok Jeong, Minwook Jeong, Jaeky Oh, Yongwoo Lee
and Young-Sik Kim*

Department of Electrical Engineering and Computer Science, DGIST, Daegu, Republic of Korea
ysk@dgist.ac.kr
LG Electronics Inc., Seoul, Republic of Korea
Inha University, Incheon, Republic of Korea

Abstract. Evaluation keys represent a primary memory and initialization bottleneck in matrix-native fully homomorphic encryption (FHE). In the Gentry–Lee (GL) framework, each Trace product yields a four-component ciphertext whose BigSwitch procedure requires two extended-ring keys, dominated by a massive product-secret key ($s_X s_Y \rightarrow s_X$). We present *Trace-Factored BigSwitch* (TFB), which structurally eliminates this product-secret evaluation key by exploiting the rank-one tensor structure of the Trace-generated key, $(1, s_X, s_Y, s_X s_Y) = (1, s_Y) \otimes (1, s_X)$. By routing the product secret through an $s_Y \rightarrow s_X$ switch followed by standard base-ring $s_X^2 \rightarrow s_X$ relinearization, TFB achieves an exact evaluation-key memory saving of $(n-1)/(2n)$ ($\approx 50\%$) without ever generating or storing $\text{evk}_{XY \rightarrow X}$. To amortize TFB’s recurring base-ring relinearization cost in blocked matrix multiplication, we introduce **FRee-L** (Fused Relinearization with noise reduction). **FRee-L** accumulates K Trace products componentwise in the four-component domain and invokes BigSwitch only once per output tile, reducing post-processing noise injections from K to one and amortizing the latency overhead as $O(1/K)$. In our OpenFHE-linked prototype ($n = 256$), TFB reduces the coefficient-domain BigSwitch key footprint from 768.0 to 385.5 MiB (49.80%), peak RSS by 15.31%, and cold key preparation by 51–63%. On real GPT-2 attention kernels ($K = 4$), **FRee-L** lowers TFB’s paired overhead to 5.90%, which further diminishes to 0.1933% ($K = 160$) and 0.0618% ($K = 512$) in deep accumulation workloads, confirming that substantial key savings are achieved with negligible workload-level compute penalty.

Keywords: Fully homomorphic encryption · matrix-native FHE · key switching · relinearization · evaluation-key memory · encrypted attention

*Young-Sik Kim is the corresponding author.

1 Introduction

Evaluation keys are a first-order engineering constraint in high-performance FHE: they consume memory capacity, must be serialized and prepared before evaluation, and are repeatedly traversed by key-switching kernels. These costs are especially visible in matrix-native FHE, where a single logical switching key may be indexed over an extended multivariate ring. In our $n = 256$ GL prototype, the evaluator-ready key material for the direct BigSwitch alone occupies 1.5 GiB. In this paper, we investigate whether this key material can be structurally eliminated—rather than merely compressed—while maintaining negligible computational overhead at the workload level.

Gentry and Lee proposed a matrix-native FHE representation in which encrypted matrix multiplication is a Trace contraction over a multivariate Gaussian-integer ring [GL25]. After public variable preparation, multiplying two ordinary ciphertexts produces four Trace blocks that decrypt as

$$C_{00} + C_{10}s_X + C_{01}s_Y + C_{11}s_Xs_Y$$

under the key vector $(1, s_X, s_Y, s_Xs_Y)$. A direct BigSwitch returns this object to a two-component ciphertext under s_X by switching both the s_Y and s_Xs_Y terms. The resulting key set contains two keys over the extended ring R'_q : $\text{evk}_{Y \rightarrow X}$ and the product-secret key $\text{evk}_{XY \rightarrow X}$. Since R'_q has an additional Y dimension relative to the base ring R_q , the product-secret key is n times larger than an ordinary R_q -level relinearization key.

Trace-Factored BigSwitch. The BigSwitch input is not an arbitrary four-component ciphertext. Its key has the rank-one tensor form

$$(1, s_X, s_Y, s_Xs_Y) = (1, s_Y) \otimes (1, s_X).$$

TFB first removes the Y -secret dependence with the existing $s_Y \rightarrow s_X$ switch. The product-secret term then becomes a degree-two term under the target secret and is handled by ordinary $s_X^2 \rightarrow s_X$ relinearization. Consequently, TFB never generates, stores, or accesses $\text{evk}_{XY \rightarrow X}$. If one extended-ring key occupies M bytes, the BigSwitch-specific key material changes from

$$M_{\text{Direct}} = 2M \quad \text{to} \quad M_{\text{TFB}} = M + \frac{M}{n},$$

which gives the exact saving $(n-1)/(2n)$ under the same decomposition and representation.

A related reorganization appears in Park’s large-matrix CCMM algorithm [Par25]: ciphertext matrix transposes produce a quadratic expression in the original secret, followed by row-wise relinearization. TFB instead targets GL’s Trace-generated key and the elimination of its extended-ring product-secret key. Section 8 explains this relationship and the different key structures.

The memory–computation trade-off. The removed product-secret key is replaced by one base-ring relinearization. In a microbenchmark restricted to the BigSwitch post-processing stage—that is, excluding GL Trace multiplication and four-component accumulation—Direct takes 2.979 s whereas TFB takes 4.181 s, a 40.36% increase. This is the standalone switching-layer penalty of TFB, not the overhead of a complete encrypted matrix-multiplication kernel. TFB is therefore a capacity-first transformation rather than a standalone latency optimization. The resulting systems question is whether the approximately fixed extra relinearization can be incurred once per accumulation group rather than for each partial product.

One output tile of a blocked matrix product is the sum of K partial Trace products. We introduce **FRee-L**—*Fused Relinearization with noise reduction*—for the GL four-component

domain. Rather than relinearizing or BigSwitching every partial product, FRee-L retains them as Ct4 objects, accumulates them componentwise, and invokes Direct or TFB post-processing once for the output tile. Thus the number of post-processing calls and associated post-processing noise injections falls from K to one. Rather than reducing the per-invocation latency of TFB.Switch, FRee-L amortizes this fixed cost by minimizing the invocation frequency, while preserving the accumulation of input-encryption and Trace noise inherent to the K products. If $T_{\text{shared}}(K)$ is the common Trace-and-accumulation prefix and Δ_{post} is TFB’s additional post-processing time, the relative overhead is

$$R(K) = \frac{\Delta_{\text{post}}}{T_{\text{shared}}(K) + T_{D,\text{post}}}.$$

The absolute delta remains, but the relative overhead decreases as the number of accumulated products grows.

Implementation and measurements. We implement an OpenFHE-linked, DCRTPoly-backed GL prototype with custom quotient-ring arithmetic, RNS/NTT kernels, an exact fast Trace, Direct BigSwitch, TFB, and deferred four-component accumulation. An independent quotient-ring oracle, the schoolbook implementation, and the fast Trace agree coefficientwise; the TFB evaluator records no product-secret-key event. At $n = 256$, TFB reduces coefficient-domain and evaluator-ready BigSwitch key footprints by 49.80%, lowers variant-isolated peak RSS by 15.31%, and shortens cold key preparation by 51–63%. Our evaluation follows the construction’s trade-off in that order: correctness, saved key state and initialization, the isolated relinearization cost, and finally its workload-level amortization through FRee-L.

For an application-level check, we evaluate post-softmax attention-value multiplication $O = PV$. Unlike the isolated post-processing microbenchmark, these kernel-level timings include the K Trace products and the schedule-specific accumulation and post-processing. On real GPT-2 activations [RWC⁺19] with $K = 4$, FRee-L reduces TFB’s paired overhead from 15.27% under eager processing to 5.90%. We additionally measure one-output-tile kernels at $K = 160$ and $K = 512$ by mapping configured maximum context lengths to the number of partial products. These ratios characterize single-tile kernel workloads parameterized by modern LLM sequence lengths, demonstrating that the relative overhead diminishes as accumulation depth increases.

Scope. We study TFB as a BigSwitch transformation for the rank-one key produced by GL Trace multiplication, using an OpenFHE-linked GL prototype and the concrete parameter set established in [GL25]. The application evaluation targets encrypted post-softmax attention-value multiplication $O = PV$: GPT-2 uses real activations, while the Qwen3 and Llama configurations determine K for one-output-tile kernel measurements.

Contributions.

1. **Product-key-free BigSwitch and exact capacity law.** We exploit the tensor structure of the Trace-generated key to replace $s_X s_Y \rightarrow s_X$ switching with $s_Y \rightarrow s_X$ switching followed by standard $s_X^2 \rightarrow s_X$ relinearization. We prove correctness for the GL conjugate embedding and derive the exact two-key footprint ratio $2n/(n+1)$.
2. **FRee-L: Fused Relinearization with noise reduction.** We introduce a GL-specific execution technique that accumulates K partial Trace products as one four-component accumulation group and performs Direct or TFB post-processing only once. We state its applicability condition, prove semantic correctness, establish that post-processing noise injections are reduced from K to one, and formulate an asymptotic relative-overhead model showing how the constant post-processing delta is effectively amortized.

3. **Verified cryptographic implementation.** We build an OpenFHE-linked custom GL/RNS/NTT prototype, implement materialized and fused TFB paths, and validate the exact Trace against an independent quotient-ring oracle. The implementation never generates or uses a product-secret evaluation key in the TFB path.
4. **Systems evaluation.** We measure key payload, evaluator-ready footprint, process RSS, cold preparation, isolated post-processing, real GPT-2 $O = PV$, and large- K one-tile kernels. The eager-versus-FRee-L ablation isolates the scheduling gain, while the large- K evaluation keeps model mappings and composed benchmark statistics explicit.

Organization. Section 2 defines the GL Trace output and the BigSwitch cost setting. Section 3 presents TFB and its exact key-size law. Section 4 introduces FRee-L and its correctness, noise, applicability, and cost model. Section 5 summarizes correctness, noise, and security. Sections 6 and 7 describe the implementation and experiments. Section 8 positions the work, and Section 9 concludes.

2 Background and BigSwitch Setting

We use a decryption-equation abstraction for RLWE-based FHE, following the standard key-switching and relinearization viewpoint of BGV, BFV, and CKKS [BGV12, FV12, CKKS17]. The multivariate ring and trace interface follow the Gentry–Lee matrix-FHE construction [GL25]. Our construction manipulates only these decryption equations and is therefore independent of whether the underlying instantiation is exact or approximate.

2.1 Rings, dimensions, and embeddings

Let n be a power of two and p an odd integer with cyclotomic polynomial $\Phi_p(W)$. For an odd modulus q , write $\mathbb{Z}_q[i] = \mathbb{Z}_q[T]/\langle T^2 + 1 \rangle$. All rings are commutative. The Y -free base ring is

$$R_q = \mathbb{Z}_q[i][X, W]/\langle X^n - i, \Phi_p(W) \rangle,$$

and the matrix-native extended ring is

$$R'_q = \mathbb{Z}_q[i][X, Y, W]/\langle X^n - i, Y^n - i, \Phi_p(W) \rangle.$$

Thus R'_q is a free R_q -module with basis $1, Y, \dots, Y^{n-1}$, and every $F \in R'_q$ has a unique expansion $F = \sum_{j=0}^{n-1} F_j Y^j$ with $F_j \in R_q$. The coefficient dimensions over $\mathbb{Z}_q[i]$ are

$$\dim_{\mathbb{Z}_q[i]}(R_q) = n\varphi(p), \quad \dim_{\mathbb{Z}_q[i]}(R'_q) = n^2\varphi(p),$$

so $\dim_{\mathbb{Z}_q[i]}(R'_q) = n \dim_{\mathbb{Z}_q[i]}(R_q)$. This additional factor n is central to the footprint analysis. The Gentry–Lee construction obtains this ring after a Gaussian-integer reindexing that removes conjugate symmetry; we adopt the convention above and note where the reindexing matters.

Let $s_X = s(X, W) \in R_q$ be the secret key, viewed in R'_q through the natural inclusion. The second ciphertext carries the same underlying secret in the Y dimension through a public *conjugate embedding* $\iota_Y : R_q \rightarrow R'_q$. On the Gaussian coefficient ring and the matrix variables, the GL preparation acts as

$$\iota_Y(i) = -i, \quad \iota_Y(X) = Y^{-1}, \quad \iota_Y(W) = W^{-1},$$

and we set $s_Y = \iota_Y(s_X) = \bar{s}(Y^{-1}, W^{-1})$, where the bar denotes Gaussian conjugation of coefficients. The relation $X^n = i$ is preserved because $(Y^{-1})^n = i^{-1} = -i = \iota_Y(i)$.

Thus ι_Y is a $\mathbb{Z}_q$ -algebra embedding (semilinear over $\mathbb{Z}_q[i]$), not a $\mathbb{Z}_q[i]$ -algebra map fixing i . The two secrets are not independently sampled; they are public images of one secret polynomial.

2.2 Coefficient representatives and norms

For $a \in \mathbb{Z}_q$, let $[a]_q$ denote its centered representative in $(-q/2, q/2]$, applied to real and imaginary parts independently for $\mathbb{Z}_q[i]$, and coefficientwise in the monomial basis for R_q or R'_q . We use the coefficient infinity norm $\|F\|_\infty = \max_c \max\{|\operatorname{Re}([c]_q)|, |\operatorname{Im}([c]_q)|\}$ over Gaussian-integer coefficients c , and the Euclidean norm where convenient. For each ring $S \in \{R_q, R'_q\}$ there is a multiplication constant $\kappa(S)$ with $\|FG\|_\infty \leq \kappa(S)\|F\|_\infty\|G\|_\infty$ for centered representatives, accounting for coefficient growth under reduction modulo the defining relations. Our results use only a fixed norm convention applied identically to the direct baseline and to TFB.

2.3 Ciphertexts as decryption equations

Let $S \in \{R_q, R'_q\}$ and let $\tau = (\tau_0, \dots, \tau_{r-1}) \in S^r$ be a key vector with $\tau_0 = 1$. An r -component ciphertext $c \in S^r$ decrypts under τ to a message representative M with error e if $\sum_j c_j \tau_j = M + e \pmod{q}$ and e is small. For two-component ciphertexts under s_X we write $\operatorname{Dec}_{s_X}(c_0, c_1) = c_0 + c_1 s_X$, and for three-component ciphertexts under powers of s_X we write $\operatorname{Dec}_{s_X}^{(2)}(d_0, d_1, d_2) = d_0 + d_1 s_X + d_2 s_X^2$. For the four-component trace product we use the key vector $(1, s_X, s_Y, s_X s_Y)$; a tuple $(C_{00}, C_{10}, C_{01}, C_{11}) \in (R'_q)^4$ decrypts to $C_{00} + C_{10} s_X + C_{01} s_Y + C_{11} s_X s_Y$. The first index records the power of s_X , the second the power of s_Y .

2.4 Auxiliary variable and trace

Matrix multiplication is expressed as a trace contraction over an auxiliary variable Z . Let $R''_q = \mathbb{Z}_q[i][X, Y, Z, W] / \langle X^n - i, Y^n - i, Z^n - i, \Phi_p(W) \rangle$, with the unique expansion $F = \sum_{k=0}^{n-1} F_k(X, Y, W) Z^k$. The normalized trace projection is $\operatorname{Tr}_Z(F) = F_0$; any global normalization of a concrete encoding is absorbed into the encoding convention. Extend the monomial coefficient norm and multiplication constant of Section 2.2 to R''_q . Under this convention, $\|\operatorname{Tr}_Z(F)\|_\infty \leq \lambda_Z \|F\|_\infty$ with $\lambda_Z = 1$. If $G \in R'_q$ is Z -free then $\operatorname{Tr}_Z(GF) = G \operatorname{Tr}_Z(F)$; in particular s_X and s_Y are Z -free and commute with Tr_Z .

After GL's public variable preparation, denote the first prepared components by $b, a \in R''_q$ and the second by $\beta, \alpha \in R''_q$. The second preparation incorporates the conjugate embedding of Section 2.1, which entails non-trivial coefficient conjugation rather than a formal variable renaming $X \mapsto Y$. Here, m and μ denote the prepared decryption phases, defined as $m = b + a s_X$ and $\mu = \beta + \alpha s_Y$. The component names below refer to these already prepared objects. Using Z -freeness of the secrets,

$$\operatorname{Tr}_Z(m\mu) = \operatorname{Tr}_Z(b\beta) + s_X \operatorname{Tr}_Z(a\beta) + s_Y \operatorname{Tr}_Z(b\alpha) + s_X s_Y \operatorname{Tr}_Z(a\alpha).$$

Defining the four trace blocks

$$C_{00} = \operatorname{Tr}_Z(b\beta), \quad C_{10} = \operatorname{Tr}_Z(a\beta), \quad C_{01} = \operatorname{Tr}_Z(b\alpha), \quad C_{11} = \operatorname{Tr}_Z(a\alpha),$$

the trace product decrypts as $C_{00} + C_{10} s_X + C_{01} s_Y + C_{11} s_X s_Y$. This is the structured four-component ciphertext relinearized by TFB.

2.5 Gadget decomposition

Let $B \geq 2$ be a gadget base and $\ell = \lceil \log_B q \rceil$, with gadget vector $g = (1, B, \dots, B^{\ell-1})$. For $S \in \{R_q, R'_q\}$, a gadget decomposition $\text{Decomp}_{B,\ell}^S : S \rightarrow S^\ell$ returns digits $(D_0, \dots, D_{\ell-1})$ with $C = \sum_h D_h B^h \pmod{q}$ and each D_h taken from a small digit set, for example centered base- B digits. The map is generally not additive because of digit carries and is not a ring homomorphism. For a coefficientwise realization extended to R'_q , selecting the Z^0 coefficient commutes with decomposition. Consequently, due to the non-linearity of carry propagation, the decomposition of an accumulated coefficient cannot be substituted by the sum of individual decompositions.

2.6 Key switching and relinearization

Let $S \in \{R_q, R'_q\}$, let $\tau \in S$ be a source secret expression, and let $s_X \in S$ be the target. An evaluation key for switching τ to s_X is a list $\text{evk}_{\tau \rightarrow X} = \{(u_h, v_h)\}_{h=0}^{\ell-1} \in (S^2)^\ell$ with

$$u_h + v_h s_X = B^h \tau + \varepsilon_h \pmod{q},$$

generated by sampling v_h uniformly, ε_h small, and setting $u_h = -v_h s_X + B^h \tau + \varepsilon_h$. Because the source secrets in TFB are algebraically related to the target secret, standard RLWE alone does not establish the pseudorandomness of these keys; Section 5.2 formalizes the precise joint auxiliary-input assumption required for the security reduction. For $C \in S$ with $\text{Decomp}_{B,\ell}^S(C) = (D_0, \dots, D_{\ell-1})$, the key-switching output is

$$\text{KS}_{\tau \rightarrow X}^S(C) = \left(\sum_h D_h u_h, \sum_h D_h v_h \right),$$

and writing $e_{\tau \rightarrow X}(C) = \sum_h D_h \varepsilon_h$ gives $U_0 + U_1 s_X = C\tau + e_{\tau \rightarrow X}(C) \pmod{q}$. Relinearization is the special case in which the source is a power of the target; ordinary relinearization uses $\text{evk}_{X^2 \rightarrow X}$ with $W_0 + W_1 s_X = C s_X^2 + e_{X^2 \rightarrow X}(C)$ for $C \in R_q$.

In TFB the coefficient to be relinearized lies in R'_q . Since R'_q is a free R_q -module, the R_q -level relinearization key extends coefficientwise: for $C = \sum_j C_j Y^j$,

$$\text{KS}_{X^2 \rightarrow X}^{R'_q}(C) = \sum_{j=0}^{n-1} Y^j \text{KS}_{X^2 \rightarrow X}^{R_q}(C_j),$$

which reuses one Y -free relinearization key across all slices. The key material is indexed over R_q even though the input lies in R'_q .

2.7 Switching keys and size notation

The direct baseline uses two R'_q -wide keys, $\text{evk}_{Y \rightarrow X} = \text{evk}_{s_Y \rightarrow s_X}$ and the product-secret key $\text{evk}_{XY \rightarrow X} = \text{evk}_{s_X s_Y \rightarrow s_X}$. TFB uses $\text{evk}_{Y \rightarrow X}$ over R'_q and $\text{evk}_{X^2 \rightarrow X} = \text{evk}_{s_X^2 \rightarrow s_X}$ over the Y -free ring R_q , extended coefficientwise; it does not use $\text{evk}_{XY \rightarrow X}$. Writing $|S|_{\text{coef}}$ for the coefficient dimension over $\mathbb{Z}_q[i]$, we have $|R_q|_{\text{coef}} = n\varphi(p)$ and $|R'_q|_{\text{coef}} = n^2\varphi(p)$; ignoring constant factors, an evaluation key over S with gadget length ℓ has footprint $\Theta(\ell |S|_{\text{coef}})$.

3 The Trace-Factored BigSwitch

This section presents the construction. We first record the tensor structure of the trace-product key, then the factorization lemma, then its validity for the conjugate embedding produced by the matrix-FHE multiplication, and finally the algorithm and its correctness. All equations are over R'_q modulo q unless stated otherwise.

3.1 The BigSwitch problem and the direct baseline

After trace multiplication the evaluator holds $C = (C_{00}, C_{10}, C_{01}, C_{11}) \in (R'_q)^4$ under $(1, s_X, s_Y, s_X s_Y)$.

Definition 1 (BigSwitch). A BigSwitch procedure is an algorithm $\text{BigSwitch} : (R'_q)^4 \rightarrow (R'_q)^2$ that, on input C , outputs (E_0, E_1) with

$$E_0 + E_1 s_X = C_{00} + C_{10} s_X + C_{01} s_Y + C_{11} s_X s_Y + e_{\text{BS}} \pmod{q},$$

where $e_{\text{BS}} \in R'_q$ is within the decryption radius of the underlying parameter set.

The direct baseline treats C as an unstructured ciphertext and applies $(P_0, P_1) = \text{KS}_{Y \rightarrow X}^{R'_q}(C_{01})$ and $(Q_0, Q_1) = \text{KS}_{XY \rightarrow X}^{R'_q}(C_{11})$, returning $(C_{00} + P_0 + Q_0, C_{10} + P_1 + Q_1)$. It is correct but requires the product-secret key $\text{evk}_{XY \rightarrow X}$, which is naturally indexed over R'_q and induces a large evaluation-key stream.

3.2 The tensor-product key structure

Lemma 1 (Tensor-product key identity). *For component order $(C_{00}, C_{10}, C_{01}, C_{11})$, the trace-product key is*

$$(1, s_X, s_Y, s_X s_Y) = (1, s_Y) \otimes (1, s_X),$$

where

$$(a_0, a_1) \otimes (b_0, b_1) = (a_0 b_0, a_0 b_1, a_1 b_0, a_1 b_1).$$

Proof. By the tensor ordering, $(1, s_Y) \otimes (1, s_X) = (1, s_X, s_Y, s_Y s_X)$. Since R'_q is commutative, $s_Y s_X = s_X s_Y$, giving the claim. $\square$

The product-secret term arises because the trace product contains one X -secret contribution and one Y -secret contribution. Because the trace-product key is rank-one as a tensor product, direct product-secret switching is unnecessary: one can first eliminate the Y -secret dependence and subsequently relinearize the remaining X -secret square.

3.3 From product-secret switching to degree-two relinearization

Lemma 2 (Product-secret switching by factorization). *Let $C \in R'_q$, let $(V_0, V_1) = \text{KS}_{Y \rightarrow X}^{R'_q}(C)$ with $V_0 + V_1 s_X = C s_Y + e_{Y \rightarrow X}(C)$, and let $(W_0, W_1) = \text{KS}_{X^2 \rightarrow X}^{R'_q}(V_1)$ with $W_0 + W_1 s_X = V_1 s_X^2 + e_{X^2 \rightarrow X}(V_1)$. Then $(W_0, V_0 + W_1)$ is a two-component ciphertext under s_X whose decryption equals*

$$C s_X s_Y + s_X e_{Y \rightarrow X}(C) + e_{X^2 \rightarrow X}(V_1).$$

In particular $C s_X s_Y$ is switched to s_X without an evaluation key for $s_X s_Y$.

Proof. We compute

$$\begin{aligned} W_0 + (V_0 + W_1) s_X &= V_0 s_X + (W_0 + W_1 s_X) \\ &= V_0 s_X + V_1 s_X^2 + e_{X^2 \rightarrow X}(V_1) \\ &= s_X (V_0 + V_1 s_X) + e_{X^2 \rightarrow X}(V_1) \\ &= C s_X s_Y + s_X e_{Y \rightarrow X}(C) + e_{X^2 \rightarrow X}(V_1). \end{aligned}$$

The evaluator forms $V_0 s_X + V_1 s_X^2$ by placing the public coefficients V_0, V_1 in the degree-one and degree-two positions; it does not multiply by the secret. Only the quadratic term $V_1 s_X^2$ requires switching, which reduces directly to standard base-ring relinearization. $\square$

The use of ordinary relinearization after reorganizing a matrix-ciphertext product also appears in Park's transpose-based CCMM [Par25, Section 4.2]. Here the factorization specifically replaces GL's $s_X s_Y \rightarrow s_X$ key by the Y -free $s_X^2 \rightarrow s_X$ key.

Algorithm 1 TFB.Switch: factorized BigSwitch using $\text{evk}_{Y \rightarrow X}$ and $\text{evk}_{X^2 \rightarrow X}$, without $\text{evk}_{XY \rightarrow X}$.

Require: trace blocks $(C_{00}, C_{10}, C_{01}, C_{11}) \in (R'_q)^4$; keys $\text{evk}_{Y \rightarrow X}, \text{evk}_{X^2 \rightarrow X}$; base B ; length ℓ

Ensure: two-component ciphertext $(E_0, E_1) \in (R'_q)^2$ under s_X

- 1: $(U_0, U_1) \leftarrow \text{KS}_{Y \rightarrow X}^{R'_q}(C_{01})$ ▷ switch the linear Y -secret term
- 2: $(V_0, V_1) \leftarrow \text{KS}_{Y \rightarrow X}^{R'_q}(C_{11})$ ▷ prepare the product-secret term
- 3: $D_0 \leftarrow C_{00} + U_0$; $D_1 \leftarrow C_{10} + U_1 + V_0$; $D_2 \leftarrow V_1$
- 4: $(W_0, W_1) \leftarrow \text{KS}_{X^2 \rightarrow X}^{R'_q}(D_2)$ ▷ ordinary relinearization
- 5: **return** $(E_0, E_1) \leftarrow (D_0 + W_0, D_1 + W_1)$

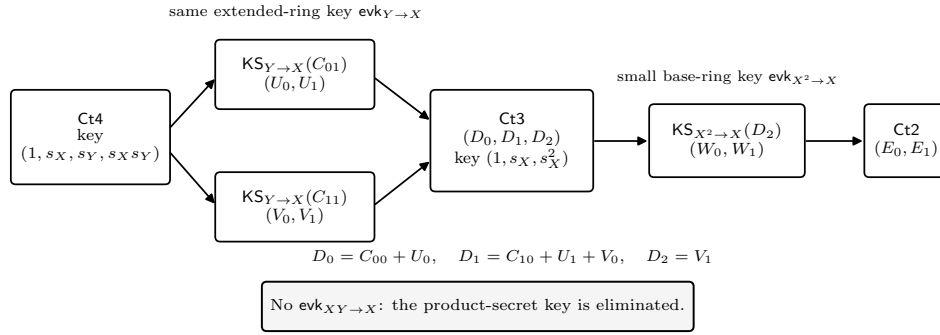


Figure 1: Dataflow of TFB.Switch. Both Y -dependent blocks use the same extended-ring $\text{evk}_{Y \rightarrow X}$; the resulting degree-two term is handled by the n -times-smaller base-ring $\text{evk}_{X^2 \rightarrow X}$. No product-secret key $\text{evk}_{XY \rightarrow X}$ is required.

3.4 Validity for the GL conjugate embedding

The factorization proof uses s_Y only as a Z -free ring element and as the source of $\text{evk}_{Y \rightarrow X}$. It therefore remains valid for the public conjugate embedding used by GL; the quotient relations are preserved because the embedding maps $i \mapsto -i$, $X \mapsto Y^{-1}$, and $W \mapsto W^{-1}$.

Lemma 3 (Compatibility with the GL conjugate embedding). *Let $\iota_Y : R_q \rightarrow R'_q$ be the public $\mathbb{Z}_q$ -algebra conjugate embedding induced by $i \mapsto -i$, $X \mapsto Y^{-1}$, and $W \mapsto W^{-1}$, and set $s_Y = \iota_Y(s_X)$. Suppose the GL multiplication produces $(C_{00}, C_{10}, C_{01}, C_{11}) \in (R'_q)^4$ decrypting as $C_{00} + C_{10}s_X + C_{01}s_Y + C_{11}s_X s_Y$. Then TFB.Switch with $\text{evk}_{Y \rightarrow X} = \text{evk}_{s_Y \rightarrow s_X}$ and $\text{evk}_{X^2 \rightarrow X}$ satisfies Theorem 1.*

Proof. The TFB derivation requires only commutativity, the two key-switching identities with source s_Y , the ordinary s_X^2 relinearization identity, and Z -freeness of s_Y . All hold by construction of ι_Y and $\text{evk}_{s_Y \rightarrow s_X}$. No coefficient-norm isometry is required: reduction of W^{-1} modulo $\Phi_p(W)$ can change the norm in the chosen monomial basis. $\square$

3.5 The algorithm

TFB applies the factorization to the two Y -dependent trace blocks, $C_{01}s_Y$ and $C_{11}s_X s_Y$, both using the same key $\text{evk}_{Y \rightarrow X}$; the second additionally uses $\text{evk}_{X^2 \rightarrow X}$. The relinearization step utilizes an R_q -level key extended coefficientwise over the Y -basis; hence, the key material remains Y -free even though the target ciphertext resides in R'_q .

3.6 Correctness

Theorem 1 (Correctness of TFB). *Let $(E_0, E_1) = \text{TFB.Switch}(C_{00}, C_{10}, C_{01}, C_{11})$, and define the key-switching errors by $U_0 + U_1 s_X = C_{01} s_Y + e_{Y \rightarrow X}(C_{01})$, $V_0 + V_1 s_X = C_{11} s_Y + e_{Y \rightarrow X}(C_{11})$, and $W_0 + W_1 s_X = V_1 s_X^2 + e_{X^2 \rightarrow X}(V_1)$. Then*

$$\begin{aligned} E_0 + E_1 s_X &= C_{00} + C_{10} s_X + C_{01} s_Y + C_{11} s_X s_Y \\ &\quad + e_{Y \rightarrow X}(C_{01}) + s_X e_{Y \rightarrow X}(C_{11}) + e_{X^2 \rightarrow X}(V_1) \pmod{q}. \end{aligned}$$

Hence TFB is a correct BigSwitch procedure whenever the total error $e_{\text{TFB}} = e_{Y \rightarrow X}(C_{01}) + s_X e_{Y \rightarrow X}(C_{11}) + e_{X^2 \rightarrow X}(V_1)$ lies within the decryption radius.

Proof. By the algorithm, $E_0 = D_0 + W_0$ and $E_1 = D_1 + W_1$ with $D_0 = C_{00} + U_0$, $D_1 = C_{10} + U_1 + V_0$, $D_2 = V_1$. Therefore

$$\begin{aligned} E_0 + E_1 s_X &= C_{00} + U_0 + (C_{10} + U_1 + V_0) s_X + (W_0 + W_1 s_X) \\ &= C_{00} + C_{10} s_X + (U_0 + U_1 s_X) + V_0 s_X + (W_0 + W_1 s_X). \end{aligned}$$

Substituting $W_0 + W_1 s_X = V_1 s_X^2 + e_{X^2 \rightarrow X}(V_1)$,

$$E_0 + E_1 s_X = C_{00} + C_{10} s_X + (U_0 + U_1 s_X) + s_X(V_0 + V_1 s_X) + e_{X^2 \rightarrow X}(V_1).$$

Substituting the two $Y \rightarrow X$ identities and using commutativity,

$$\begin{aligned} E_0 + E_1 s_X &= C_{00} + C_{10} s_X + C_{01} s_Y + e_{Y \rightarrow X}(C_{01}) \\ &\quad + s_X(C_{11} s_Y + e_{Y \rightarrow X}(C_{11})) + e_{X^2 \rightarrow X}(V_1) \\ &= C_{00} + C_{10} s_X + C_{01} s_Y + C_{11} s_X s_Y \\ &\quad + e_{Y \rightarrow X}(C_{01}) + s_X e_{Y \rightarrow X}(C_{11}) + e_{X^2 \rightarrow X}(V_1), \end{aligned}$$

which is the claimed equation. $\square$

The statement holds for any tuple under $(1, s_X, s_Y, s_X s_Y)$; in the matrix-FHE construction such tuples arise exactly from trace multiplication, so TFB applies at the level of the relinearization equation, independent of matrix size or message encoding.

3.7 Product-key-free property and semantic equivalence

Definition 2 (Product-key-free BigSwitch). A BigSwitch procedure is *product-key-free* if its key generation never generates, publishes, or uses an evaluation key whose source secret is $s_X s_Y$.

Corollary 1 (TFB is product-key-free). *TFB is product-key-free. Its evaluation-key set is $\{\text{evk}_{Y \rightarrow X}, \text{evk}_{X^2 \rightarrow X}\}$ and excludes $\text{evk}_{XY \rightarrow X}$.*

Proof. Algorithm 1 contains only $\text{KS}_{Y \rightarrow X}^{R'_q}$ and $\text{KS}_{X^2 \rightarrow X}^{R'_q}$ calls; the latter uses the R_q -level relinearization key extended coefficientwise. No call to $\text{KS}_{XY \rightarrow X}^{R'_q}$ occurs, and no key for $s_X s_Y$ is generated. $\square$

Corollary 2 (Semantic equivalence to the direct baseline). *Applied to the same trace-product tuple, the direct baseline and TFB decrypt to the same plaintext $C_{00} + C_{10} s_X + C_{01} s_Y + C_{11} s_X s_Y \pmod{q}$, up to their respective key-switching errors.*

Proof. The direct decryption equation is derived in Section 3.1; the TFB equation is Theorem 1. Both have the same noiseless term. $\square$

3.8 Evaluation-key footprint

Corollary 3 (Footprint reduction). *For the BigSwitch-specific key material, TFB replaces the R'_q -wide product-secret key $\text{evk}_{XY \rightarrow X}$ by the R_q -wide relinearization key $\text{evk}_{X^2 \rightarrow X}$. The replaced component is smaller by the factor $|R'_q|_{\text{coef}}/|R_q|_{\text{coef}} = n$, and for the two-key BigSwitch set the idealized footprint ratio is*

$$\frac{2\ell |R'_q|_{\text{coef}}}{\ell |R'_q|_{\text{coef}} + \ell |R_q|_{\text{coef}}} = \frac{2n}{n+1}.$$

Proof. Immediate from $|R'_q|_{\text{coef}} = n |R_q|_{\text{coef}}$ substituted into the ratio. $\square$

We emphasize that Corollary 3 characterizes the evaluation-key storage rather than execution latency; TFB incurs one additional base-ring relinearization, whose standalone latency and workload-level amortization are evaluated in Section 7. The exact saving is $\text{Saving}(n) = (n-1)/(2n)$, approaching 50% as n grows.

4 FRee-L: Fused Relinearization with Noise Reduction

TFB changes the post-processing of one trace-generated four-component ciphertext. Matrix multiplication adds a second opportunity: a single output tile is normally the sum of several partial products, and post-processing can be deferred until after this sum. We call the technique FRee-L, short for *Fused Relinearization with noise reduction*. It fuses the K per-product post-processing/relinearization calls into one call on a componentwise Ct4 sum. “Noise reduction” refers only to the post-processing noise terms: K such injections become one, while the K Trace and input-encryption errors still accumulate.

4.1 Blocked matrix products and four-component accumulation

Consider one output tile of a blocked matrix product,

$$Z_{r,c} = \sum_{k=1}^K A_{r,k} B_{k,c}.$$

For each inner-tile index k , the GL trace product returns

$$C^{(k)} = (C_{00}^{(k)}, C_{10}^{(k)}, C_{01}^{(k)}, C_{11}^{(k)}) \in (R'_q)^4.$$

The eager schedule post-processes every $C^{(k)}$ into a two-component ciphertext and then adds the outputs. FRee-L instead forms the componentwise sum

$$\overline{C}_{ij} = \sum_{k=1}^K C_{ij}^{(k)} \pmod{q}$$

and invokes the selected Direct or TFB post-processing path once on $\overline{C}$.

Let Post_D denote the direct BigSwitch and Post_{TFB} the TFB switch. The two schedules are

$$E_P^{\text{eager}} = \sum_{k=1}^K P(C^{(k)}),$$

$$E_P^{\text{FRee-L}} = P\left(\sum_{k=1}^K C^{(k)}\right), \quad P \in \{\text{Post}_D, \text{Post}_{\text{TFB}}\}.$$

While these ciphertexts are not necessarily bitwise identical due to carry propagation during gadget decomposition, they decrypt to the identical noiseless matrix-tile sum.

Algorithm 2 FRee-L accumulation for one output tile: retain K partial products as Ct4 and post-process once.

Require: partial products $C^{(1)}, \dots, C^{(K)} \in (R'_q)^4$

Require: post-processing path $P \in \{\text{Post}_D, \text{Post}_{\text{TFB}}\}$

```

1:  $\overline{C}_{00}, \overline{C}_{10}, \overline{C}_{01}, \overline{C}_{11} \leftarrow 0$ 
2: for  $k = 1, \dots, K$  do
3:   for  $(i, j) \in \{0, 1\}^2$  do
4:      $\overline{C}_{ij} \leftarrow \overline{C}_{ij} + C_{ij}^{(k)} \pmod{q}$ 
5:   end for
6: end for
7: return  $P(\overline{C}_{00}, \overline{C}_{10}, \overline{C}_{01}, \overline{C}_{11})$ 

```

Applicability. An accumulation group is valid when all K partial products contribute only through addition to the same downstream output and no intervening operation requires an individually post-processed two-component ciphertext. This condition holds for the inner-product accumulation of a blocked matrix-multiplication output tile.

Proposition 1 (Semantic correctness of deferred post-processing). *Suppose a post-processing path P satisfies*

$$\text{Dec}_{s_X}(P(C)) = \text{Dec}_4(C) + e_P(C)$$

for every valid four-component input C . Assume that all partial tuples share the modulus, key vector, ciphertext slot layout, and encoding scale, and that their aggregate is also valid under the chosen decoding budget. Then

$$\text{Dec}_{s_X}(E_P^{\text{FRee-L}}) = \sum_{k=1}^K \text{Dec}_4(C^{(k)}) + e_P\left(\sum_{k=1}^K C^{(k)}\right).$$

Hence FRee-L computes the same noiseless sum as the eager schedule for both Direct and TFB.

Proof. The four-component decryption equation is linear in the ciphertext components, so $\text{Dec}_4(\sum_k C^{(k)}) = \sum_k \text{Dec}_4(C^{(k)})$. Applying the correctness equation of P to the aggregate gives the claim. The direct path satisfies the premise by the standard key-switch equations, and TFB by Theorem 1. $\square$

Corollary 4 (Post-processing noise injections). *If a uniform single-call bound $\|e_P(C)\|_\infty \leq \eta_P$ holds for all valid inputs, then the post-processing contribution is bounded by $K\eta_P$ under eager processing and by η_P under FRee-L. This statement concerns the post-processing contribution only; the trace-product and input-encryption errors still accumulate across the K partial products.*

Proof. The eager output contains $\sum_k e_P(C^{(k)})$, whose norm is at most $K\eta_P$ by the triangle inequality. Proposition 1 shows that the deferred output contains one post-processing error term. $\square$

4.2 Cost model and vanishing relative overhead

Let T_{Trace} be the time for one partial Trace product; let T_{Ct2Add} and T_{Ct4Add} be two- and four-component additions; and let $T_{P,\text{post}}$ be the cost of $P \in \{\text{Post}_D, \text{Post}_{\text{TFB}}\}$. Eager processing and FRee-L cost

$$\begin{aligned}
T_P^{\text{eager}}(K) &= KT_{\text{Trace}} + KT_{P,\text{post}} + (K-1)T_{\text{Ct2Add}}, \\
T_P^{\text{FRee-L}}(K) &= KT_{\text{Trace}} + (K-1)T_{\text{Ct4Add}} + T_{P,\text{post}}.
\end{aligned}$$

Thus FFree-L changes the number of post-processing calls from K to one, trading two-component additions for four-component additions. For the Direct-versus-TFB comparison under FFree-L, define

$$T_{\text{shared}}(K) = KT_{\text{Trace}} + (K - 1)T_{\text{Ct4Add}}.$$

Then $T_D(K) = T_{\text{shared}}(K) + T_{D,\text{post}}$ and $T_{\text{TFB}}(K) = T_{\text{shared}}(K) + T_{\text{TFB},\text{post}}$. Writing $\Delta_{\text{post}} = T_{\text{TFB},\text{post}} - T_{D,\text{post}}$, the relative overhead is

$$R(K) = \frac{\Delta_{\text{post}}}{T_{\text{shared}}(K) + T_{D,\text{post}}}.$$

Proposition 2 (Amortization law). *If Δ_{post} is bounded independently of K and $T_{\text{Trace}} > 0$, then $R(K) \rightarrow 0$ as $K \rightarrow \infty$. More precisely, $R(K) = O(1/K)$ when the per-product Trace and addition costs remain bounded away from zero and infinity.*

Proof. The numerator is independent of K , whereas the denominator grows linearly in K . $\square$

This is an asymptotic cost-model statement; it does not assert decodability for unbounded K at a fixed parameter set. Importantly, Proposition 2 characterizes asymptotic latency amortization rather than asserting $T_{\text{TFB}}(K) < T_D(K)$; the absolute post-processing delta Δ_{post} persists across all K , but its relative overhead becomes negligible as the shared Trace accumulation dominates execution time.

4.3 Attention-value multiplication

For post-softmax self-attention, one head computes $O_h = P_h V_h$. When the sequence dimension L is blocked into tiles of side length n , one output tile accumulates

$$K = \left\lceil \frac{L}{n} \right\rceil$$

partial encrypted matrix products. This is exactly the schedule above. Our real-activation experiment focuses on the core homomorphic $O = PV$ tensor contraction from GPT-2, targeting the primary linear-algebra bottleneck rather than ancillary pipeline stages (such as projections, masking, and homomorphic softmax). For the Qwen3 and Llama configurations in Section 7, the context lengths are mapped to the accumulation depth K for benchmark profiling without loading full model weights.

5 Correctness, Noise, and Security

We state the correctness, noise, and security guarantees needed for the construction and evaluation. The central algebraic reductions are proved in the paper; the security theorem is presented as a standard hybrid reduction under the explicit evaluation-key assumption below.

5.1 Trace-and-TFB correctness and noise

Let the two prepared inputs decrypt as $b + as_X = M_0 + e_0$ and $\beta + \alpha s_Y = M_1 + e_1$, and define

$$M_{\text{prod}} = \text{Tr}_Z(M_0 M_1), \quad e_{\text{trace}} = \text{Tr}_Z(M_0 e_1) + \text{Tr}_Z(M_1 e_0) + \text{Tr}_Z(e_0 e_1).$$

Let $\Delta_{\text{dec}} > 0$ denote the parameter-dependent decoding error budget determined by the plaintext encoding, message representative, and admissible range conditions, rather than an implicit bound inferred merely from centering modulo q . Combining the trace-product equation with Theorem 1 gives the following statement.

Theorem 2 (Trace-and-TFB correctness). *The TFB output (E_0, E_1) satisfies*

$$E_0 + E_1 s_X = M_{\text{prod}} + e_{\text{trace}} + e_{\text{TFB}} \pmod{q},$$

where

$$e_{\text{TFB}} = e_{Y \rightarrow X}(C_{01}) + s_X e_{Y \rightarrow X}(C_{11}) + e_{X^2 \rightarrow X}(V_1).$$

Decryption succeeds whenever $\|e_{\text{trace}} + e_{\text{TFB}}\|_\infty < \Delta_{\text{dec}}$.

Proof (Proof sketch). The trace product decrypts to $M_{\text{prod}} + e_{\text{trace}}$. Applying Theorem 1 to its four components adds exactly e_{TFB} . $\square$

Let Δ_B bound the centered gadget digits, let η_Y and η_{X^2} bound the two evaluation-key errors, and let γ_X be the coefficient-norm operator bound for multiplication by the Y -free secret s_X . Standard gadget-switch analysis gives the compact worst-case bound

$$\|e_{\text{TFB}}\|_\infty \leq (1 + \gamma_X) \ell\kappa(R'_q) \Delta_B \eta_Y + \ell\kappa(R_q) \Delta_B \eta_{X^2}. \quad (1)$$

Together with

$$\|e_{\text{trace}}\|_\infty \leq \lambda_Z \kappa(R''_q) (\|M_0\|_\infty \|e_1\|_\infty + \|M_1\|_\infty \|e_0\|_\infty + \|e_0\|_\infty \|e_1\|_\infty),$$

Equation (1) yields a sufficient correctness condition by requiring the sum of the two bounds to remain below Δ_{dec} . While TFB does not reduce cumulative ciphertext noise, our empirical evaluation verifies correct decoding and sufficient noise headroom up to $K = 512$. Under FRee-L, the post-processing contribution is injected once after accumulation rather than once per partial product, while input and Trace errors still accumulate with K .

5.2 Security preservation

TFB leaves the GL setup, public-key encryption, and decryption algorithms unchanged and modifies only the public BigSwitch key tuple. Let

$$(s_X, \text{Pub}_0) \leftarrow \text{Setup}_0(1^\lambda)$$

denote the common GL setup, where Pub_0 contains the public key, public parameters, and every other secret-dependent public object shared by Direct and TFB. Set $s_Y = \iota_Y(s_X)$ and sample the two TFB-specific keys, with independent key-generation randomness, as

$$K_Y \leftarrow \text{GKey}_{R'_q}(s_X, s_Y), \quad K_2 \leftarrow \text{GKey}_{R_q}(s_X, s_X^2).$$

Let (U_Y, U_2) be independent uniform encodings with exactly the same rings, gadget rows, RNS limbs, dimensions, and serialization formats as (K_Y, K_2) .

Definition 3 (Joint auxiliary-input evaluation-key game). For $\beta \in \{0, 1\}$, experiment $\text{Exp}_\beta^{\text{J-EKIND}}(\mathcal{D})$ samples (s_X, Pub_0) as above and gives $\mathcal{D}$ either

$$(\text{Pub}_0, K_Y, K_2) \quad \text{if } \beta = 0, \quad (\text{Pub}_0, U_Y, U_2) \quad \text{if } \beta = 1.$$

It outputs the bit returned by $\mathcal{D}$, and

$$\text{Adv}_{\text{TFB}}^{\text{J-EKIND}}(\mathcal{D}) = \left| \Pr[\text{Exp}_0^{\text{J-EKIND}}(\mathcal{D}) = 1] - \Pr[\text{Exp}_1^{\text{J-EKIND}}(\mathcal{D}) = 1] \right|.$$

Assumption 3 (Joint related-secret evaluation-key indistinguishability). *For every PPT distinguisher $\mathcal{D}$, the advantage in Definition 3 is negligible.*

The keys are challenged jointly because both depend on the same s_X and are revealed together with all common auxiliary material. Independent marginal games that isolate each key individually are insufficient. Assumption 3 is an explicit circular/related-secret assumption and is not implied by the plain-LWE estimator below.

Let $\Pi_{\text{GL-TFB}}$ publish (Pub_0, K_Y, K_2) and use TFB.Switch as its public evaluator. Its IND-CPA experiment runs

$$\begin{aligned} (m_0, m_1, \text{st}) &\leftarrow \mathcal{A}_1(\text{Pub}_0, K_Y, K_2), \quad |m_0| = |m_1|, \\ b &\leftarrow \{0, 1\}, \quad c^* \leftarrow \text{Enc}_{\text{pk}}(m_b), \\ b' &\leftarrow \mathcal{A}_2(\text{st}, c^*), \end{aligned}$$

and outputs $[b' = b]$. We use the convention $\text{Adv}^{\text{IND-CPA}} = |\Pr[b' = b] - 1/2|$. The base experiment Π_{GL}^0 is identical but omits the two TFB-specific key streams.

Theorem 4 (Security of GL with TFB). *For every PPT IND-CPA adversary $\mathcal{A}$, there exist efficient $\mathcal{B}_0, \mathcal{B}_1$ such that*

$$\text{Adv}_{\Pi_{\text{GL-TFB}}}^{\text{IND-CPA}}(\mathcal{A}) \leq \text{Adv}_{\Pi_{\text{GL}}^0}^{\text{IND-CPA}}(\mathcal{B}_0) + \text{Adv}_{\text{TFB}}^{\text{J-EKIND}}(\mathcal{B}_1).$$

No public hint for the product secret $s_X s_Y$ occurs in the construction.

Proof. Game G_0 is the real TFB IND-CPA experiment. In G_1 , replace (K_Y, K_2) jointly by (U_Y, U_2) while leaving Pub_0 unchanged. A distinguisher for Definition 3 runs $\mathcal{A}_1$, samples the challenge bit, encrypts with the public key contained in Pub_0 , runs $\mathcal{A}_2$, and outputs one exactly when $\mathcal{A}$ guesses correctly. Hence the change in success probability is bounded by $\text{Adv}_{\text{TFB}}^{\text{J-EKIND}}(\mathcal{B}_1)$. In G_1 , the two key streams are independent of the secret and challenge bit; $\mathcal{B}_0$ samples them locally around its base-GL challenge and perfectly simulates G_1 . The triangle inequality gives the stated bound. $\square$

The theorem formalizes the exact joint auxiliary-input assumption needed by TFB; it does not convert product-key removal into additional security bits. The materialized and fused evaluators apply deterministic public arithmetic to the same key tuple and therefore induce the same security experiment. The concrete LWE security evaluation in Section 6.2 estimates attack cost but does not establish Assumption 3, full GL/RLWE security, or circular security.

6 Implementation and Experimental Setup

6.1 Implementation scope

We implement an OpenFHE-linked, DCRTPoly-backed GL prototype. OpenFHE 1.5.1 is used for evaluation-key storage, serialization, and multi-modulus representation, while the GL quotient-ring arithmetic, Trace, Direct BigSwitch, TFB, and FRee-L schedule are implemented explicitly. All compared variants use the same RNS/NTT backend, build, parameters, and input artifacts within each comparison. The primary timings run on an AMD Threadripper PRO 5995WX with one physical core pinned for the timed region. Our measurements are consequently those of an *OpenFHE-linked custom GL prototype*, not a production **CryptoContext**-native GL implementation. The artifact bundle contains a table-to-file manifest and a host-capture script for the OS/kernel, compiler and flags, prototype and OpenFHE revisions, CPU topology, and affinity settings.

6.2 Parameter Selection and Concrete Security Evaluation

The primary measurements use the parameter set in Table 1. A SageMath 10.7/lattice-estimator evaluation (plain-LWE dimension $N = 8192$, full ternary secret, and $\sigma = 3.2$

error) gives B1 ciphertext estimates of 174.62/158.51/202.45 bits and limiting B2 evaluation-key estimates of 107.30/97.52/137.72 bits under ADPS16 classical, ADPS16 quantum, and estimator-default MATZOV, respectively. Direct and TFB obtain the same optimized B2 attack costs because the selected attacks use only about 2^{13} samples, confirming that product-secret-key elimination does not alter standard LWE attack complexity. While these plain-LWE estimates do not capture non-standard circular-security aspects, they establish concrete hardness under standard lattice attacks; the estimator script and configuration are provided with the artifact.

Table 1: Representative GL prototype parameters matching [GL25] used for primary $n = 256$ measurements.

Parameter	Value
Matrix dimension n	256
Cyclotomic/ W parameter p	17
Ciphertext modulus Q	3 RNS limbs, approximately 141 bits
Auxiliary modulus q_0	approximately 59 bits
Decomposition	3 modulus digits plus q_0
W transform	Rader-17
Error sampling	tail-cut discrete Gaussian, $\sigma = 3.2$
CPU	AMD Threadripper PRO 5995WX
Primary timing	one pinned physical core
OpenFHE	release 1.5.1; DCRTPoly-backed storage
Security evaluation	SageMath 10.7; lattice-estimator <code>3e48ef4</code>
Backend	custom GL/RNS/NTT prototype

6.3 Exact fast Trace

Because a naive schoolbook Trace incurs prohibitive latency for large- K experiments, we implement a fast exact Trace exploiting the quotient-ring structure and Rader-17 transforms. Validation compares three independent paths coefficientwise: a standalone reference oracle derived directly from the algebraic quotient relations, the schoolbook Trace, and the fast Trace. Gaussian conjugation is mathematically essential; omitting it invalidates the ring homomorphism across all 20 targeted test instances.

At $n = 256$, each full four-component output contains 25,165,824 checked coefficients. Two independent seeds produce zero mismatches between the oracle, schoolbook, and fast implementations. The baseline fast Trace execution takes 4.3919 s, while an optimized zero-allocation implementation achieves 4.3747 s; both measurements are reported to distinguish the standard setup from the zero-allocation variant.

6.4 Post-Processing and Scheduling Design Space

We separate three independent choices. The post-processing path is Direct or TFB; the accumulation schedule is eager or FRee-L; and the kernel realization is materialized or intra-post-process fused. All variants share the same Trace implementation, RNS/NTT backend, deterministic decomposition, underlying secret-key context, and ciphertext inputs; their evaluation-key sets are the respective Direct or TFB sets defined above. The fused TFB realization completes each Trace coefficient, applies the same decomposition and key multiply-adds as the materialized path, and streams the shared $Y \rightarrow X$ key once across both Y -dependent blocks. It yields identical ciphertexts up to modular addition order, strictly preserving standard coefficient-wise decomposition semantics.

6.5 Key representations and measurement boundaries

We report two evaluation-key representations: a serialized coefficient-domain payload and an evaluator-ready form containing the transformed values and precomputation required by the common backend. Key-payload measurements exclude unrelated ciphertext working sets, whereas process peak RSS includes the ciphertext buffers and backend tables shared by both variants. These representations affect the measured capacity and initialization cost but do not change the TFB construction.

6.6 Measurement and statistical methodology

All primary CPU comparisons use one pinned physical core and the same binary, parameters, underlying secret-key context, and ciphertext inputs within a comparison, with the evaluation keys required by each variant. The baseline fast Trace uses three warm-up executions followed by 20 measured trials; the post-processing-only comparison uses 20 single-core measured trials; and cold-key preparation uses 10 independent trials with no warm-cache phase, each verified cold with `0_DIRECT` and `mincore`. Standalone execution times are marginal medians. Peak RSS is a variant-isolated process measurement rather than a key-payload statistic.

For aligned Direct/TFB records $(D_i, T_i)_{i=1}^N$, define

$$d_i = T_i - D_i, \quad r_i = \frac{T_i - D_i}{D_i}.$$

Columns labelled *paired delta* and *paired overhead* report the arithmetic means $N^{-1} \sum_i d_i$ and $N^{-1} \sum_i r_i$, respectively. Displayed Direct and TFB times remain marginal medians, so their simple difference or ratio need not equal the paired statistic. The GPT-2 ablation reuses identical encrypted inputs and rotates the four variant orders to reduce execution-order bias. The reported 42.21% and 54.76% eager slowdowns reflect relative speedups between the two accumulation schedules, distinct from the paired Direct-versus-TFB overheads.

Each maximum- K pair contains a measured shared Trace/accumulation prefix and measured Direct/TFB post-processing components. We form

$$T_{D,i} = T_{\text{shared},i} + T_{D,\text{post},i}, \quad T_{\text{TFB},i} = T_{\text{shared},i} + T_{\text{TFB},\text{post},i}$$

before computing d_i and r_i . Thus the reported totals are *synthesized from measured components* rather than projected values or single continuous timer runs; continuous outer-timer runs serve only as drift checks. To provide a direct and deterministic characterization of the kernel runtimes across trials, we report paired point estimates (sample means and medians). Measured and analytical values are labelled separately, key payloads use MiB, and OS RSS uses decimal MB.

6.7 Attention workloads

The real-activation workload captures GPT-2 [RWC⁺19] layer-5 post-softmax tensors and evaluates one layer’s $O = PV$ kernel at sequence length 1024, with 12 query/KV heads, head dimension 64, and 12 heads batched in a 16-slot W axis. The 1024 inner dimension gives $K = 4$ at $n = 256$. The implementation uses 12-bit quantization for P and V and encoder scale $\Delta = 2^{44}$; the plaintext $P_h V_h$ reference agrees with the model’s pre- W_O head output with a relative error of approximately 10^{-7} .

For the maximum- K study, model names only determine the context-length mapping in Table 7. GPT-2 uses real activations in the separate experiment above. For the Qwen3 and Llama configurations, the benchmarks evaluate single output-tile GL kernels parameterized by the respective model context lengths rather than loading full model weights.

7 Experimental Evaluation

The evaluation follows the progression introduced in Section 1. We first verify that TFB preserves the GL Trace result without a product-secret evaluation key. We then quantify the reductions in BigSwitch key capacity, process memory, and cold key preparation, followed by the recurring post-processing cost introduced by the additional base-ring relinearization. Finally, we evaluate how effectively FRee-L’s deferred K -way accumulation amortizes this cost in matrix workloads. Statistical conventions follow Section 6.6.

7.1 Correctness and fast-Trace validation

The comprehensive test suite successfully passes all 31 regression checks across 12 test suites, with zero memory and undefined-behavior violations verified via ASan, UBSan, and LeakSanitizer. The $n = 256$ noisy Direct and TFB paths both decode successfully, confirming that no product-secret evaluation key is accessed during TFB execution. Independent Trace validation covers exhaustive small- n basis and boundary monomials, random dense inputs, and two full $n = 256$ coefficientwise comparisons against the schoolbook path.

Each full $n = 256$ comparison examines 25,165,824 coefficients and finds zero mismatches. The fast Trace is $480.2\times$ and $488.1\times$ faster than schoolbook for the two independent seeds. These checks establish the correctness of the exact fast Trace used by the large- K experiments.

7.2 Key capacity, process memory, and cold preparation

We distinguish two levels of memory reduction: the BigSwitch-specific evaluation-key payload and the memory footprint of the complete process. At $n = 256$, the measured key sizes exactly match $M_{\text{Direct}} = 2M$ and $M_{\text{TFB}} = M + M/n$. Figure 2 visualizes the resulting size law, while Table 2 reports the measured $n = 256$ payloads and process RSS.

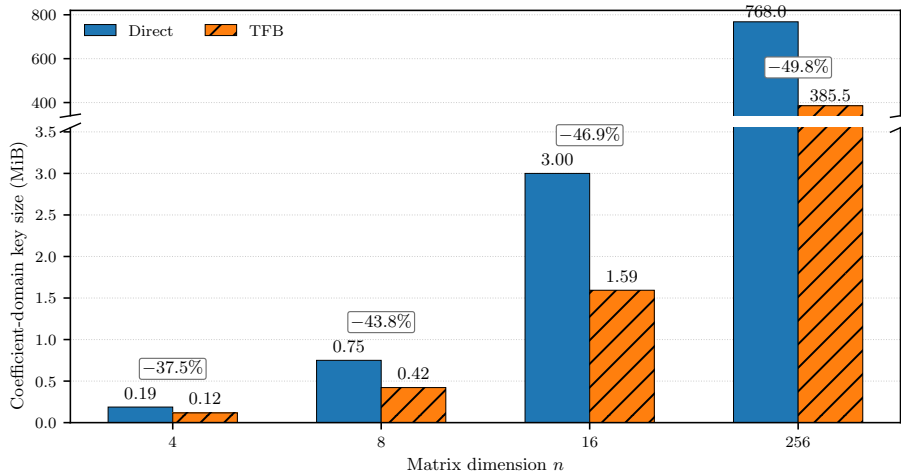


Figure 2: BigSwitch-specific coefficient-domain key size as n grows. The $n = 256$ values are measured; the $n = 4, 8, 16$ points are analytical values obtained from $M_{\text{Direct}} = 2M$ and $M_{\text{TFB}} = M + M/n$, anchored to the measured $n = 256$ layout. The saving approaches 50%.

Table 2: Measured key and process-memory footprint at $n = 256$. Key payloads use MiB; OS peak RSS uses decimal MB.

Representation	Direct	TFB	Reduction
Coefficient-domain key	768.0 MiB	385.5 MiB	49.80%
Evaluator-ready key	1536.0 MiB	771.0 MiB	49.80%
Variant-isolated peak RSS	4763 MB	4034 MB	15.31%

Both key representations achieve a 49.80% reduction, while process peak RSS decreases by 15.31% because Direct and TFB retain the same ciphertext working set and backend tables; we therefore keep key payload and whole-process memory as separate claims.

Table 3: Median cold evaluation-key preparation over ten `0_DIRECT/mincore`-verified trials at $n = 256, p = 17$.

Operation	Direct (ms)	TFB (ms)	Reduction
Coefficient read + deserialize	556.1	207.1	62.76%
Evaluator-ready read + deserialize	1321.8	641.3	51.48%
Coefficient to evaluator-ready	4623.4	2248.9	51.36%

The compact evaluation key provides its most pronounced advantage in one-shot cold-start evaluations. In the measured cold-start crossover, TFB is 14.2% faster for one evaluation, nearly equal at two, and slower from three evaluations onward. The corresponding break-even model is $r^* = 2.061$, where the key-reuse count r is distinct from the FFree-L accumulation factor K . These results establish a capacity and initialization gain; the next subsection isolates the recurring compute cost incurred for this capacity gain.

7.3 Post-processing-only cost of TFB

Table 4 isolates the BigSwitch post-processing layer. The timed region excludes GL Trace generation and four-component accumulation. Replacing the extended product-secret key with a base-ring key requires one additional $s_X^2 \rightarrow s_X$ relinearization, increasing the one-call post-processing time from 2.979 s to 4.181 s, or 40.36%.

Table 4: Median BigSwitch post-processing-only time over 20 single-core trials at $n = 256, p = 17$; GL Trace and four-component accumulation are excluded.

Path	Time (s)	Decode
Direct BigSwitch (two extended keys)	2.979	PASS
TFB fused	4.181	PASS
Isolated base-ring relinearization	1.000	PASS
TFB post-processing-only overhead	40.36%	

The 40.36% ratio is therefore a switching-layer quantity: both its numerator and denominator contain only BigSwitch post-processing. TFB is primarily a capacity-oriented optimization: it halves the key footprint and accelerates cold-start preparation, while trading off standalone post-processing latency. FFree-L operates at the scheduling level: rather than modifying the latency of an individual TFB.Switch invocation, it incurs the base-ring relinearization cost only once across an entire K -way accumulation group. The

workload-level ratios below use a broader denominator that also includes the shared Trace computation and accumulation.

7.4 Real GPT-2 attention-value multiplication

We now move from the switching-only microbenchmark to the target accumulation pattern. The primary application experiment evaluates real GPT-2 layer-5 activations for one post-softmax $O = PV$ kernel at sequence length $L = 1024$ ($K = 4$), isolating the core attention-value multiplication from the remainder of the self-attention pipeline. Unlike Table 4, each reported kernel time includes all four Trace products and the schedule-specific path: eager per-product post-processing followed by two-component addition, or FRee-L four-component accumulation followed by one final post-processing call. The four variants use the same encrypted inputs and are interleaved by balanced rotation.

Table 5: Real GPT-2 $O = PV$ schedule ablation at $L = 1024, K = 4$. Kernel times include the four Trace products and the schedule-specific accumulation/post-processing path; displayed times sum the measured group means of four output tiles, while delta and overhead are arithmetic means over paired Direct/TFB records.

Schedule	Direct (s)	TFB (s)	Delta (s)	TFB overhead
FRee-L	79.608	84.305	4.697	5.90%
Eager	113.207	130.469	17.262	15.27%

Using the displayed aggregate schedule times, eager execution is 42.21% slower than FRee-L for Direct and 54.76% slower for TFB. These speedups stem from the deferred accumulation strategy of FRee-L, independent of the underlying post-processing variant. Within the same kernel-level measurement, FRee-L also lowers TFB’s overhead relative to Direct from 15.27% to 5.90%, a factor of about 2.6. While standalone post-processing incurs a 40.36% microbenchmark penalty, the workload-level overhead is amortized to 5.90% because the shared Trace computation dominates the execution time and FRee-L invokes post-processing only once per four partial products. Every evaluation path achieves zero decoded-integer mismatches against the quantized reference, confirming exact homomorphic arithmetic fidelity.

A separate one-output-tile scaling run on real GPT-2 activations gives the trend in Table 6. The overhead for $K = 4$ in Table 6 (6.88%) differs slightly from the multi-tile ablation result in Table 5 (5.90%) due to differences in benchmark scheduling and trial sets between the isolated single-tile run and the full-schedule experiment.

Table 6: Real GPT-2 one-output-tile scaling with FRee-L. Each row uses checkpoint activations; overhead is the arithmetic mean over paired Direct/TFB records.

L	K	Direct (s)	TFB (s)	Paired overhead
256	1	7.517	8.951	19.79%
512	2	12.073	13.433	12.15%
1024	4	21.090	22.440	6.88%

7.5 Large- K scaling with model-mapped one-tile kernels

Having established the scheduling effect with real GPT-2 activations, we next measure how the same fixed post-processing delta scales with larger accumulation factors. As in the GPT-2 kernel experiment, the reported overhead is workload-level rather than

switching-only: its denominator includes the shared K -product Trace prefix in addition to one final post-processing call. Table 7 maps an official configured sequence-length value L to the number of $n = 256$ inner tiles [RWC⁺19, Qwe25, Met24]. Only the GPT-2 study above uses checkpoint activations; the Qwen3 and Llama rows are one-output-tile kernel mappings.

Table 7: Target model configurations and the derived accumulation depth $K = \lceil L/n \rceil$ for single-tile evaluation.

Model mapping	Configured L	n	$K = \lceil L/n \rceil$
GPT-2 Small	1,024	256	4
Qwen3 14B config	40,960	256	160
Llama 3.1 8B	131,072	256	512

For large K , an approximately one-second post-processing difference is smaller than the drift between independent multi-minute runs. These percentages are not directly comparable to the isolated 40.36% microbenchmark, as the denominator here includes the substantial shared Trace prefix in addition to post-processing. We therefore measure the shared Trace prefix and both post-processing components, then compose the totals according to Section 4.2. Table 8 reports these synthesized execution totals. Displayed totals are marginal medians; delta and ratio are paired means and need not equal the simple difference or ratio of the displayed medians.

Table 8: One-output-tile benchmark totals synthesized from measured components. Direct/TFB totals are marginal medians; delta and ratio denote arithmetic means over paired component records.

Mapping	K	Direct (s)	TFB (s)	Paired delta (s)	Paired ratio
GPT-2 Small	4	21.254	22.748	1.501	6.9313%
Qwen3 14B	160	732.235	733.496	1.294	0.1933%
Llama 3.1 8B	512	2325.505	2327.070	1.438	0.0618%

The paired delta remains approximately 1.29–1.50 s while the common Trace prefix grows. Correspondingly, $KR(K)$ is approximately 0.2773, 0.3093, and 0.3164 for $K = 4, 160, 512$, consistent with the asymptotic $O(1/K)$ amortization model established in Section 4.2. All paths decode correctly with zero integer mismatches, require no additional modulus limb, and retain 26.0, 23.3, and 22.0 bits of noise headroom, respectively.

Continuous end-to-end wall-clock runs are monitored as an empirical baseline to verify timing consistency. At $K = 160$, approximately 0.26% drift over a 731-second run exceeds the 0.1933% effect being measured, so the continuous and composed ratios are not statistically interchangeable. Accordingly, we report these sub-percent results as composed benchmark overheads evaluated alongside the shared Trace prefix.

8 Related Work

Relinearization and matrix FHE. BGV, BFV, and CKKS use key switching to return multiplication ciphertexts to two components [BGV12, FV12, CKKS17, LPR13]. Rather than serving as an arbitrary key-switching primitive, TFB specifically factors the structured $s_X s_Y$ term produced by GL Trace multiplication through $Y \rightarrow X$ switching followed by standard $X^2 \rightarrow X$ relinearization. This also differs from multi-key FHE [LATV12], because

$s_Y = \iota_Y(s_X)$ is a public image of one owner’s secret rather than an independently sampled user key.

Transpose-based CCMM and relinearization. Park [Par25] reduces large-matrix CCMM to four plaintext matrix multiplications and three ciphertext matrix transposes (C-MTs) in a coefficient-encoded RLWE representation. In Section 4.2 (p. 19 of the ePrint version), C-MT reorganizes the left- and right-secret factors so that the product has the form

$$MM' \approx H_2 \text{Toep}(s^2) + H_1 \text{Toep}(s) + H_0,$$

where s is Park’s RLWE secret and H_0, H_1, H_2 denote the resulting intermediate matrices, renamed here for clarity. The remaining quadratic term is removed by a final row-wise $s^2 \rightarrow s$ relinearization stage, followed by rescaling. This means one relinearization per output-row ciphertext, not a single key-switch invocation for the entire matrix: Algorithm 3 uses N such switches for an $N \times N$ instance, where N is the RLWE ring degree.

TFB shares the high-level strategy of reorganizing a matrix-ciphertext product before ordinary relinearization, but acts on GL’s four-component Trace output in R'_q , rather than on a row-/column-wise RLWE bundle. It reuses $\text{evk}_{Y \rightarrow X}$ for both Y -dependent blocks and replaces $\text{evk}_{XY \rightarrow X}$ with the Y -free $\text{evk}_{X^2 \rightarrow X}$, reused across the Y -slices. Our contribution is this GL-specific product-secret-key elimination and its $2M \rightarrow M + M/n$ footprint law, not the general idea of ending a reorganized CCMM with ordinary relinearization. Park’s lightweight variants instead reduce stored automorphism keys through on-demand key updates, addressing a different key family.

Linear algebra, scheduling, and key movement. Packed FHE linear algebra uses rotations, diagonal/BSGS schedules, and hoisting [SV14, AZ26], with compiler and systems support [JVC18, DSC⁺18, DKS⁺19, CDA⁺21, BCH⁺25]. GL instead realizes ciphertext–ciphertext matrix multiplication as a native Trace operation [GL25]; TFB specifically optimizes the subsequent BigSwitch procedure. FRee-L is the GL-specific scheduling contribution: it retains partial products under $(1, s_X, s_Y, s_X s_Y)$ and post-processes once per accumulation group. This complements prior work identifying evaluation-key capacity and movement as systems bottlenecks [JLK⁺20, BKS⁺21, KKK⁺22, GVB⁺22, ZCC⁺25, WHZ⁺24, dCAY⁺21].

Encrypted attention. FHE inference combines matrix kernels with approximations of nonlinear functions [GBDL⁺16, JVC18, CHK⁺24, PMW⁺26, PPP⁺26]. In this work, our benchmark targets the core linear contraction of post-softmax attention ($O = PV$), while non-linear homomorphic softmax evaluation and GL bootstrapping [CGK⁺26] represent orthogonal, complementary components.

9 Conclusion and Outlook

TFB exploits the rank-one key produced by GL Trace multiplication to replace the large $s_X s_Y \rightarrow s_X$ key with $s_Y \rightarrow s_X$ switching and an n -times-smaller $s_X^2 \rightarrow s_X$ relinearization key. Under the same representation, BigSwitch key material decreases from $2M$ to $M + M/n$. At $n = 256$, the measured key footprint decreases by 49.80%, variant-isolated peak RSS by 15.31%, and cold key preparation by 51–63%.

This capacity reduction incurs one additional base-ring relinearization, leading to a post-processing-only overhead of 40.36%. FRee-L performs post-processing once per compatible accumulation group rather than once per partial product. On real GPT-2 post-softmax $O = PV$ with $K = 4$, it lowers TFB’s paired kernel-level overhead from 15.27% under eager processing to 5.90%. The model-mapped one-output-tile measurements at $K = 160$ and 512

further support the amortization model: the fixed additional cost becomes small relative to the shared Trace prefix. These large- K results are measured-component-composed statistics, not independent end-to-end model timings.

The two techniques therefore address complementary costs: TFB reduces the key set, while FRee-L reduces the frequency of finalization. The latter applies when compatible partial products contribute additively to the same output and remain within the aggregate decoding budget. It neither eliminates input-encryption and Trace errors nor makes the extra TFB relinearization free. The demonstrated benefit is thus a capacity-computation trade-off for accumulated matrix kernels, rather than a universal reduction in per-call latency.

Future work includes integrating these kernels with the representation conversions, scale management, nonlinear operations, and bootstrapping where needed for complete encrypted attention. This would extend the evaluation beyond the current post-softmax $O = PV$ scope. More broadly, TFB and FRee-L motivate considering evaluation-key capacity and accumulation schedules jointly when designing matrix-native FHE systems.

References

- [AZ26] Sajjad Akherati and Xinmiao Zhang. Triple-hoisted baby-step giant-step linear transformation over ckks homomorphic encryption and hardware accelerator, 2026.
- [BCH⁺25] Youngjin Bae, Jung Hee Cheon, Guillaume Hanrot, Jai Hyun Park, and Damien Stehlé. Fast homomorphic linear algebra with blas, 2025.
- [BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12*, pages 309–325. ACM, 2012.
- [BKS⁺21] Fabian Boemer, Sejun Kim, Gelila Seifu, Fillipe D. M. de Souza, and Vinodh Gopal. Intel hexl: Accelerating homomorphic encryption with intel avx512-ifma52, 2021.
- [CDA⁺21] Meghan Cowan, Deeksha Dangwal, Armin Alaghi, Caroline Trippel, Vincent T. Lee, and Brandon Reagen. Porcupine: A synthesizing compiler for vectorized homomorphic encryption, 2021.
- [CGK⁺26] Eric Crockett, Craig Gentry, Hyojun Kim, Yeongmin Lee, and Yongwoo Lee. Efficient bootstrapping in fully homomorphic encryption for matrix arithmetic. Cryptology ePrint Archive, Paper 2026/956, 2026.
- [CHK⁺24] Wonhee Cho, Guillaume Hanrot, Taeseong Kim, Minje Park, and Damien Stehlé. Fast and accurate homomorphic softmax evaluation, 2024.
- [CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT 2017*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437. Springer, 2017.
- [dCAY⁺21] Leo de Castro, Rashmi Agrawal, Rabia Yazicigil, Anantha Chandrakasan, Vinod Vaikuntanathan, Chiraag Juvekar, and Ajay Joshi. Does fully homomorphic encryption need compute acceleration?, 2021.

- [DKS⁺19] Roshan Dathathri, Blagovesta Kostova, Olli Saarikivi, Wei Dai, Kim Laine, and Madanlal Musuvathi. Eva: An encrypted vector arithmetic language and compiler for efficient homomorphic computation, 2019.
- [DSC⁺18] Roshan Dathathri, Olli Saarikivi, Hao Chen, Kim Laine, Kristin Lauter, Saeed Maleki, Madanlal Musuvathi, and Todd Mytkowicz. Chet: Compiler and runtime for homomorphic evaluation of tensor programs, 2018.
- [FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012.
- [GBDL⁺16] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 201–210. PMLR, 2016.
- [GL25] Craig Gentry and Yongwoo Lee. Fully homomorphic encryption for matrix arithmetic. Cryptology ePrint Archive, Paper 2025/1935, 2025.
- [GVBP⁺22] Robin Geelen, Michiel Van Beirendonck, Hilder V. L. Pereira, Brian Huffman, Tynan McAuley, Ben Selfridge, Daniel Wagner, Georgios Dimou, Ingrid Verbauwhede, Frederik Vercauteren, and David W. Archer. Basalisc: Programmable hardware accelerator for bgv fully homomorphic encryption, 2022.
- [JLK⁺20] Wonkyung Jung, Eojin Lee, Sangpyo Kim, Keewoo Lee, Namhoon Kim, Chohong Min, Jung Hee Cheon, and Jung Ho Ahn. Heaan demystified: Accelerating fully homomorphic encryption through architecture-centric analysis and optimization, 2020.
- [JVC18] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. Gazelle: A low latency framework for secure neural network inference, 2018.
- [KLK⁺22] Jongmin Kim, Gwangho Lee, Sangpyo Kim, Gina Sohn, John Kim, Minsoo Rhu, and Jung Ho Ahn. Ark: Fully homomorphic encryption accelerator with runtime data generation and inter-operation key reuse, 2022.
- [LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 1219–1234. ACM, 2012.
- [LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM*, 60(6):43:1–43:35, 2013.
- [Met24] Meta AI. Introducing Llama 3.1: Our most capable models to date. Official Meta AI release documentation, July 2024. Documents the 128K context length of the Llama 3.1 8B and 70B models.
- [Par25] Jai Hyun Park. Ciphertext-ciphertext matrix multiplication: Fast for large matrices. In *Advances in Cryptology – EUROCRYPT 2025*, Lecture Notes in Computer Science. Springer, 2025.
- [PMW⁺26] Hanjun Park, Byeong-Seo Min, Jiheon Woo, Min-Wook Jeong, Jongho Shin, Yongwoo Lee, Young-Sik Kim, and Yongjune Kim. Efficient softmax reformulation for homomorphic encryption via moment generating function, 2026.

- [PPP⁺26] Jaiyoung Park, Sejin Park, Jai Hyun Park, Jung Ho Ahn, Jung Hee Cheon, Guillaume Hanrot, Jung Woo Kim, Minje Park, and Damien Stehlé. Scaling up privacy-preserving ml: A ckks implementation of llama-2-7b, 2026.
- [Qwe25] Qwen Team. Qwen3-14B model card and configuration. Official Hugging Face model repository, 2025. The official model card documents the default `max_position_embeddings` value of 40,960.
- [RWC⁺19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, February 2019.
- [SV14] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic simd operations. *Designs, Codes and Cryptography*, 71(1):57–81, 2014.
- [WHZ⁺24] Zhiwei Wang, Haoqi He, Lutan Zhao, Peinan Li, Zhihao Li, Dan Meng, and Rui Hou. Chameleon: An efficient fhe scheme switching acceleration on gpus, 2024.
- [ZCC⁺25] Junxue Zhang, Xiaodian Cheng, Gang Cao, Meng Dai, Yijun Sun, Han Tian, Dian Shen, Yong Wang, and Kai Chen. Flash-fhe: A heterogeneous architecture for fully homomorphic encryption acceleration, 2025.