DUPLEX: ACTIVELY SECURE, DEALER-FREE TWO-PARTY FUNCTION SECRET SHARING FOR PRIVATE LLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Private LLM inference lets a client and a model provider evaluate a model while neither reveals its secret, and function secret sharing (FSS) is the most communication-efficient tool for its nonlinear layers, yet every deployed FSS inference system assumes a trusted dealer and semi-honest evaluators. A recent two-party FSS construction removes both assumptions with dealer-free key generation and a dynamic cross-phase verification chain, but stops short of an inference system. We present DUPLEX, which turns that verification core into an end-to-end dealer-free, actively secure two-party framework: (i) a secure runtime whose session state machine makes consuming an uncertified key or releasing an unauthenticated output structurally impossible; (ii) a cost-aware compiler that unifies lookup tables and low-degree splines for GELU, Softmax, SiLU and LayerNorm, solving for the segmentation that prices active-verification overhead under an accuracy budget δ; and (iii) a PCG/VOLE producer–consumer scheduler with Half-Tree key compression and bit-exact CUDA/CUTLASS kernels. Across six open LLM checkpoints including BERT-Base, GPT-2 and Llama-3.2-1B, the full stack executes end to end with per-token fidelity up to 0.9998 and next-token top-1 agreement up to 100.0%; a naive spline-only plan costs up to 3.2$\times$ more, and the device-resident exact ring GEMM runs 5.8$\times$ faster than the CPU. The paper completes the design with a simulation-based security proof of the full DUPLEX scheme against one statically corrupted, malicious party.

1 INTRODUCTION

A private large-language-model (LLM) inference is two parties evaluating a model on an input where the client will not reveal the prompt and the provider will not reveal the weights. The demand is concrete: clinical notes, legal discovery, source code and internal documents are exactly the material people want a model to read and exactly the material they cannot send in the clear.

The cryptographic design space is understood in outline. Homomorphic encryption handles the linear layers, oblivious transfer or garbled circuits the nonlinearities, and a decade of engineering has driven the constants down (Juvekar et al., 2018; Mishra et al., 2020; Rathee et al., 2020; Huang et al., 2022; Hao et al., 2022; Pang et al., 2024; Lu et al., 2025). Function secret sharing (FSS) has been the most effective answer for the nonlinear half: SIGMA (Gupta et al., 2024) evaluates a GPT-scale Transformer with FSS gates that move *no* online bytes, and Orca (Jawalkar et al., 2024) puts the machinery on GPUs.

The gap. Every one of those FSS systems makes one or both of two assumptions that private LLM inference cannot afford. They assume a **trusted dealer** that knows the input masks in the clear and manufactures both parties' keys; and they assume **semi-honest** evaluators. A client that will not send its prompt to the provider will not send its masks to the provider's affiliate either, and a provider whose model is its product has no basis for assuming a paying client follows the protocol. A recent two-party FSS construction (referred to throughout as the *verification core*) removes the cryptographic obstacle: it makes distributed key generation dealer-free and actively secure against one static malicious party, and proves a lifting theorem showing that every gate which is a public affine post-processing of a constant number of DPF/DCF evaluations inherits active security at no

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

online cost, using a two-plane authenticated key and a dynamic cross-phase verification chain. We integrate that core as our first contribution and build the missing system on top of it. Four gaps stand between the core and a deployable private LLM inference framework.

(G1) No dealer-free active pipeline exists for LLM execution. The core certifies a single gate; an inference is thousands of gates, linear layers, truncations and a final release. Binding all of them to one session (so that a malicious party can neither consume an uncertified key nor release an unauthenticated output) is a systems problem the core leaves open.

(G2) The approximant is priced for the wrong adversary. Every FSS system approximates transcendental operators with a mixture of tables and low-degree splines chosen under a semi-honest cost model in which a triple is free and a key is never certified. Under dealer-free active security a triple carries a sacrifice, every instance carries a full-domain certification pass and every instance carries a second authentication plane, which reprices tables against splines: a mode chosen without this accounting can cost several times more than the priced optimum.

(G3) Dealer-free preprocessing is serialised with inference. Removing the dealer turns key generation into an interactive protocol whose cost the online path pays up front. No FSS inference system decouples this dealer-free preprocessing from online execution with a producer–consumer pipeline, compresses it with Half-Tree, or accelerates it on the GPU.

(G4) There is no end-to-end evidence under a unified stronger threat model. No prior system reports an actual dealer-free, actively secure execution of a modern open LLM.

Our approach. The core insight of DUPLEX is to use the cross-phase authentication chain of the verification core as the security backbone of an inference system: precompile approximation plans and correlation inventories from the *public* model architecture, then let a background producer generate and certify consumable key material while an online consumer executes the model against a ready queue and releases only across a certification barrier. Everything that depends on the secret prompt or weights stays inside the authenticated share domain; everything that can be fixed from public information is fixed offline.

Our contributions.

- **C1: a dealer-free, actively secure, two-party LLM inference framework** that integrates the verification core through a minimal, auditable interface and binds generation, certification, evaluation and authenticated release into one session state machine (Sections 3, 5 and 7). It realises the declared ideal functionality with abort under one static malicious corruption; an injected-deviation campaign confirms that malicious deviations are caught and honest runs release (Appendix K.1).
- **C2: a cost-aware piecewise-approximation compiler** that unifies lookup tables and low-degree splines for GELU, Softmax, SiLU and LayerNorm and solves, by dynamic programming under an accuracy budget δ, for the per-region plan that jointly accounts for curvature and active-verification overhead (Section 4). It is provably optimal on the grid, closes the realised error loop by compile–measure–tighten, and avoids the up-to $3.2\times$ penalty of naive single-mode plans (Section 8.1).
- **C3: a PCG/VOLE producer–consumer preprocessing scheduler** that decouples dealer-free offline key production from online inference, with a Half-Tree compression backend and CUDA/CUTLASS kernels (Section 6). We instantiate $\mathcal{F}_{\mathrm{prep}}$ concretely from silent VOLE, prove the queue's security invariants, and measure overlap, key compression and bit-exact GPU acceleration (Section 8.2 and Appendix K.2).
- **C4: an end-to-end evaluation** on six open checkpoints (BERT-Base, GPT-2, OPT-125M, Qwen2.5-0.5B, Qwen2.5-1.5B, Llama-3.2-1B) against their real weights, with fair composite baselines organised by a five-tier security-target taxonomy, reporting cost, fidelity and the security–performance trade-off (Section 8).

A summary of results and the scope of the security claim are given in Appendix C; related work is surveyed in Appendix A.

2 BACKGROUND, SYSTEM MODEL, AND THREAT MODEL

Notation. Secrets are additively shared between two parties over an audit algebra $\mathbb{K}$, instantiated as one of $\mathbb{F}_{p^2}$ with $p = 2^{61} - 1$, $\mathrm{GF}(2^{128})$, or the Galois ring $\mathrm{GR}(2^{k+s}, d)$; the choice fixes the soundness–bandwidth trade-off of certification. Real-valued weights and activations are encoded in fixed point with format Q21.16 (one sign bit, 21 integer, 16 fractional bits). A transformer is described by its public dimensions: L layers, hidden width d, feed-forward width d_{ff}, H attention heads, and context length S. Two message authentication code (MAC) keys separate the two computation phases: Δ authenticates offline wire values and Λ authenticates FSS outputs. We write $[\![x]\!]$ for a Δ-authenticated sharing of x and $\langle\!\langle x \rangle\!\rangle$ for a Λ-authenticated sharing. Throughout, λ is the security parameter, negl denotes a negligible function, and $\not\approx$ denotes computational indistinguishability.

Function secret sharing and offset gates. A distributed point function (DPF) shares $\beta \cdot \mathbf{1}\{x = \alpha\}$ and a distributed comparison function (DCF) shares $\beta \cdot \mathbf{1}\{x < \alpha\}$, each as two compact keys whose local evaluations sum to the intended output. Nonlinear operators are expressed as *offset gates*: for a mask pair $(r_{\mathrm{in}}, r_{\mathrm{out}})$ and a base function G, the offset family is

$$G^{[r_{\mathrm{in}}, r_{\mathrm{out}}]}(\hat{x}) \;=\; G(\hat{x} - r_{\mathrm{in}}) + r_{\mathrm{out}}, \qquad \hat{x} = x + r_{\mathrm{in}},$$

where the masked input $\hat{x}$ is public while x remains hidden. Evaluating a private nonlinearity thus reduces to a public-index lookup into shared tables. This full-domain FSS-for-read pattern originates with linear-scan ORAM (Doerner & Shelat, 2017) and underlies private table and spline evaluation (Wagh, 2022; Storrier et al., 2023).

The verification core. DUPLEX treats the construction of Xue et al. (2026) as a fixed interface and does not modify it. Its central object is the two-plane authenticated key $F_{\theta,\Lambda}$: alongside the function key θ it carries a Λ-plane so that every evaluation returns a Λ-authenticated sharing $\langle\!\langle \cdot \rangle\!\rangle$ rather than a bare additive share, making authentication intrinsic to evaluation rather than a bolt-on check. These keys are produced by the dealer-free distributed generation GenTrace in the $\mathcal{F}_{\mathrm{prep}}/\mathcal{F}_{\mathrm{coin}}$ hybrid, with no third party contributing correlated randomness. A batch of keys is validated by GenCheck, whose six-residual moment ledger accepts an honest batch and rejects a tampered one except with probability $\approx 2n/|E|$, where n is the domain bit-length and $E \subseteq \mathbb{K}$ is the certification challenge set. Certified outputs are opened through a single aggregated Λ-MAC by Release, and Aggregate joins certified shares across distinct audit domains. Table 3 (appendix) summarizes this interface. Combining certified evaluations is free online, by the lifting result restated below.

Theorem 1 (Gate lifting, Xue et al., 2026)**.** *Every affine post-processing of a constant number of certified DPF/DCF evaluations is itself a Λ-authenticated offset gate and adds no online communication or certification cost beyond that of the underlying evaluations.*

Theorem 1 is the property DUPLEX exploits: once a small set of DPF/DCF evaluations is certified, any linear combination of them (the shape most operator approximations take after masking) remains authenticated for free, so cost accrues per certified evaluation rather than per gate.

System model. Two parties run the inference. A *client* holds a private prompt x; a *provider* holds private weights W. The following are public and known to both: the model architecture, all tensor shapes, the context length S, the fixed-point format, the operator catalogue, the approximation plan chosen by the compiler (Section 4), the preprocessing inventory, the producer/consumer queue policy, and the schedule. The following remain secret: the prompt, the weights, all intermediate activations, the masks $(r_{\mathrm{in}}, r_{\mathrm{out}}, \dots)$, and the MAC keys Δ and Λ. The offline *producer*, the online *consumer*, and the *verifier* are logical roles executed by these same two parties; no third-party dealer exists and no party is trusted to generate correlated randomness on behalf of the other.

Threat model. DUPLEX adopts the security definition of the verification core (Xue et al., 2026) unchanged. The adversary statically corrupts at most one of the two parties and may behave arbitrarily, including rushing (choosing its messages after seeing the honest party's within a round). The guarantee is security *with abort*: the honest party either obtains the correct output or aborts, and a malicious deviation is caught by certification before any Release. Protocols are stated in the $\mathcal{F}_{\mathrm{prep}}/\mathcal{F}_{\mathrm{coin}}$ hybrid, isolating correlated-randomness and public-coin generation into ideal functionalities. The model accommodates a narrowly scoped, adversary-chosen abort condition

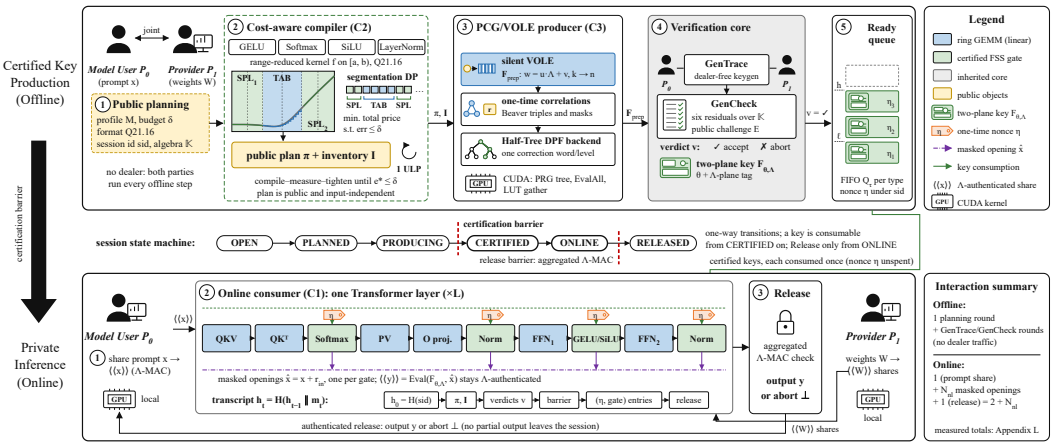


Figure 1: DUPLEX workflow. *Offline (certified key production):* a public planning round fixes the model profile M, the budget δ and the session (1); the cost-aware compiler (Section 4) turns every operator into a public LUT/spline plan π with its inventory $\mathcal{I}$ (2); the PCG/VOLE producer (Section 6) expands silent subfield-VOLE into one-time authenticated correlations (Half-Tree DPF backend, CUDA kernels) (3); the inherited verification core (Xue et al., 2026) generates keys dealer-free with GenTrace and certifies them with GenCheck (4); only keys with an accepting verdict enter the authenticated ready queue under one-time nonces η (5). The session state machine (Section 5) separates the phases at the certification barrier. *Online (private inference):* the prompt is shared once, ring GEMMs open Beaver triples, each nonlinear gate consumes one certified key on a masked opening $\hat{x} = x + r_{\text{in}}$, every artefact enters the transcript h_t, and one aggregated Λ-MAC release returns the output or aborts. The summary counts steps at gate granularity (N_{nl} nonlinear invocations); the runtime batches them per layer and also pays the GEMM openings, so measured round totals are those of Table 11. Figure 4 is the detailed version.

defined by a single predicate over the secrets. Table 4 (appendix) collects the goals, confidentiality, hybrid assumptions, and out-of-scope items.

3 DUPLEX OVERVIEW

Components. Figure 1 splits DUPLEX at the *certification barrier*. Above it, key production runs offline from public information: a public planning round fixes the model profile and budget; a *public operator compiler* (Section 4, C2) turns each transcendental operator into a public LUT/spline plan π that prices active-verification overhead; a *PCG/VOLE producer* (Section 6, C3) expands silent VOLE into one-time correlations with Half-Tree-compressed seeds; the verification core (Xue et al., 2026) generates and certifies the keys dealer-free with GenTrace/GenCheck; and only certified material enters an *authenticated ready queue*. Below the barrier an *online consumer* (Section 5, C1) executes the model: the prompt is secret-shared once, each certified nonlinear gate is evaluated from a one-time key on a masked wire, and the secret-shared ring GEMMs, which carry the Beaver openings accounted in Section 8, are offloaded to bit-exact CUDA/CUTLASS kernels; Release opens the output only across the barrier, or aborts.

The design question behind this architecture, the actors, a single-layer walk-through, the per-unit key lifecycle and the detailed version of Figure 1 (Figure 4) are given in Appendix D; Table 2 (Appendix A) positions DUPLEX against 25 prior systems.

4 COST-AWARE PIECEWISE APPROXIMATION

An inference spends most of its nonlinear budget on a small catalogue of transcendental operators, each realised as a mixture of lookup tables and low-degree polynomial splines. Semi-honest FSS picks the cheapest mixture by slope-driven heuristics and fixed segment counts, pricing only the online byte count. Under the dealer-free active setting of the verification core (Xue et al., 2026)

this prices the wrong adversary: every function-secret-sharing (FSS) payload now carries a second authentication plane, a full-domain certification pass over the audit algebra $\mathbb{K}$, and, per degree, an authenticated multiplication whose triple is sacrificed. The optimum moves, differently by operator shape; this section prices segmentation against the stronger model and proves it grid-optimal.

Why the semi-honest optimum moves. The two modes obey different error laws. For scalar $f\colon I \to \mathbb{R}$ with derivatives f', f'' and piece width h, a piecewise-*constant* table cell has worst-case error $\frac{1}{2}|f'|h$, so $h \le 2\delta/|f'|$ and the cell count over I scales as $\int_I |f'|\,\mathrm{d}x/(2\delta)$, the *total variation* of f. A *linear* spline segment has error $\frac{1}{8}|f''|h^2$, so $h \le \sqrt{8\delta/|f''|}$ and the segment count scales as $\int_I \sqrt{|f''|}\,\mathrm{d}x/\sqrt{8\delta}$, the square root of *curvature*, almost independent of slope. Hence a shape rule: a steep but nearly straight region needs many table cells but almost no spline segments, and a tightly curved region the reverse (Figure 5). Active security unflattens the per-unit cost: each spline degree adds one authenticated multiplication (a sacrificed $[\![\cdot]\!]$ triple) and a certification pass, so raising degree to cut segments is charged twice. The priced optimum keeps degree low (one or two) and leans on segmentation, by an operator-dependent margin in $\int|f'|$ and $\int\sqrt{|f''|}$ rather than a fixed rule. Figure 5 and the solver pseudocode (Figure 6) appear in Appendix E.

Definition 1 (Segmentation plan)**.** *Let $f\colon I \to \mathbb{R}$ be a scalar operator on a range-reduced interval $I = [a, b)$ in fixed point Q21.16, and $u_{\mathrm{ulp}} = 2^{-16}$ one output unit in the last place (ULP). With $Q(\cdot)$ round-to-nearest onto the Q21.16 grid, the realised error of an approximant $\hat{f}$ is $e(\hat{f}) = \max_{x\in I}\big|Q(\hat{f}(x)) - Q(f(x))\big|$ in ULP, and δ is the accuracy budget. Fix candidate breakpoints $a = x_0 < x_1 < \cdots < x_N = b$, the* grid atoms*; any $[x_i, x_j)$ with $i < j$ is a* region*, admitting two* modes*:*

- ***TAB** (table): tiled by piecewise-constant cells whose values form the payload of a distributed point function (DPF), read as an offset gate $G^{[r_{\mathrm{in}}, r_{\mathrm{out}}]}(\hat{x}) = G(\hat{x} - r_{\mathrm{in}}) + r_{\mathrm{out}}$ with $\hat{x} = x + r_{\mathrm{in}}$ the public masked input;*
- ***SPL**$_d$ (degree-d spline), $1 \le d \le D$: a single degree-d polynomial whose piecewise support is realised by a distributed comparison function (DCF) prefix, again an offset gate, with the $d{+}1$ coefficients as payload.*

A plan *π is an ordered partition $I = \bigcup_j R_j$ with a mode $m_j \in \{\mathrm{TAB}\} \cup \{\mathrm{SPL}_d : 1 \le d \le D\}$ and coefficients per region; it is δ-*feasible *if $e_{R_j} \le \delta$ on every R_j.*

Cost model. Feasibility carries the error; the objective prices resources. Both gate families are *affine* post-processings of a constant number of DPF/DCF evaluations, and each region emits a Λ-authenticated sharing $\langle\!\langle y_j \rangle\!\rangle$ vanishing outside its region, so the output is the plain sum $\langle\!\langle y \rangle\!\rangle = \sum_j \langle\!\langle y_j \rangle\!\rangle$ in the Λ plane; an "off" region contributes an authenticated zero. Composing a hybrid thus needs no oblivious selection or secure multiplexer, which is what makes a per-region table/spline mixture affordable. A region's marginal cost aggregates, in online-path units, the DPF/DCF payload in *two* authentication planes (Δ for the offline key wires, Λ for the FSS output), the full-domain certification pass over $\mathbb{K}$, the authenticated multiplications a degree-d spline spends on Horner recombination, and the key's size, dealer-free generation, and evaluation; a network term is included but does not bind. A TAB region of width w pays for $k = \lceil w/h \rceil$ cells (h from local total variation), so $c_{\mathrm{TAB}}(i, j)$ is dominated by k payload elements and one certification pass; an SPL_d region pays a fixed offset-gate cost plus $d{-}1$ authenticated multiplications and one pass, $c_{\mathrm{SPL}}(d, i, j)$. Each mode also carries a one-time *fixed bundle*: table machinery (DPF generator and certification harness), spline machinery (DCF generator and Horner triples), costing B_{TAB} and B_{SPL} once per operator, amortised over its regions.

Segmentation as a dynamic program. The fixed bundles, paid at most once each, are the only inter-region coupling; all else is additive, giving an exact dynamic program (DP). State (i, s) pairs grid atom i with a two-bit flag $s = (b_{\mathrm{TAB}}, b_{\mathrm{SPL}}) \in \{0, 1, 2, 3\}$ marking which bundles are paid; $C[i, s]$ is the minimum cost of covering $[x_i, b)$ given s. The transition picks a next breakpoint $j > i$ and mode m; if $\mathrm{err}(i, j, m) \le \delta$ it pays $c_m(i, j)$ plus any unpaid bundle for m's family, updates $s \mapsto s'$, and recurses into $C[j, s']$, base case $C[N, \cdot] = 0$, answer $C[0, \varnothing]$ (Figure 6). Since the suffix cost depends on history only through the current atom and paid bundles, an optimal plan's suffix is optimal: optimal substructure by a standard exchange argument. Figure 5 works GELU through the

solver end to end: the smooth tails clear δ with a degree-1 or degree-2 spline at almost no online cost, the curved core needs either a degree-3 spline or a table, and the table wins once its key material fits the budget, halving the online cost of the all-spline plan.

Theorem 2 (Grid optimality)**.** *Among all δ-feasible plans whose breakpoints lie on the grid $\{x_0, \ldots, x_N\}$ and whose region modes lie in $\{\mathrm{TAB}\} \cup \{\mathrm{SPL}_d : 1 \le d \le D\}$, the dynamic program of Figure 6 returns one of minimum cost. It runs in $O\big(N^2(D{+}1)\big)$ time and $O(N)$ memory, given the precomputed feasibility oracle* $\mathrm{err}(i, j, m)$.

The proof, the compile–measure–tighten loop and the range reductions are deferred to Appendices E, I and J.

5 Cross-Phase Verification and Secure Execution

Session state machine. A session is a monotone automaton over OPEN $\to$ PLANNED $\to$ PRODUCING $\to$ CERTIFIED $\to$ ONLINE $\to$ RELEASED. Its transitions fix π; start GenTrace; require an aggregate accept verdict from GenCheck (the *certification barrier*); admit activation inputs and enable key consumption; and require the aggregated Λ-MAC check (the *release barrier*). Transitions are one-directional and each gate consults only the current state, so no stage can be re-entered to substitute inputs after its verdict is recorded.

Transcript binding. A collision-resistant hash H maintains a growing session digest tying every artifact to sid. From $h_0 = \mathsf{H}(\mathsf{sid})$ the runtime folds in the plan, each verdict, the certification barrier, and the release as $h_t = \mathsf{H}(h_{t-1} \,\|\, m_t)$, where m_t ranges over π, the key-constant commitments and inventory counts, each per-bundle verdict v, the barrier markers, and every queue entry admitted to evaluation. Since a consumed key's identifier and its accept verdict both lie in the chain, an evaluation step that opened material never absorbed under this sid, or a bundle whose verdict is absent, yields a digest the honest party will not match and the session aborts. The digest thus witnesses that plan, inventory, key constants, ready-queue entries, and executed model belong to one session.

The certification barrier. A key may be consumed only after its own accept verdict is recorded, i.e., only in CERTIFIED or ONLINE. The scheduler of §6 enforces this as a queue invariant: no uncertified key enters the ready queue, consumption is one-time, and refill depends only on public counts, so an adversary cannot advance a key past the barrier by manipulating timing. A gate whose bundle is still PRODUCING is refused rather than deferred.

Algorithm 1 formalises the framework as one protocol, Π_{Duplex}: it fixes the public plan, produces and certifies key material in the background, executes the model on authenticated shares across the certification barrier, and ends in a single aggregated Release. The per-layer online loop is expanded as Algorithm 3 in Appendix I.

Algorithm 1, the stage interfaces, the masked-opening handoff, the authentication invariant and the release barrier are detailed in Appendix F; the security guarantee is stated in §7.

6 Producer–Consumer Preprocessing and Hardware Acceleration

Removing the dealer relocates its work: dealer-free generation makes key manufacture an interactive protocol the two parties run *and certify* themselves (the GenTrace primitive and GenCheck predicate of the verification core (Xue et al., 2026) yield a verdict before any key is used), so on demand its latency lands on the online critical path. *Can dealer-free, certified FSS key generation be decoupled from online evaluation so that its cost is paid off the critical path, and can the cheap primitives underneath it (silent correlations, compressed keys, GPU kernels) be supplied without weakening the guarantees the online path depends on?* Contribution C3 answers in four parts: instantiating $\mathcal{F}_{\mathrm{prep}}$ from silent subfield-VOLE; a producer–consumer inventory that certifies key material ahead of demand and hands it across a certification barrier; a Half-Tree backend shrinking the read-only keys; and CUDA/CUTLASS kernels running the arithmetic bit-exactly on a GPU. The *physical* parties are P_0, P_1; "producer" and "consumer" are *logical* roles each runs concurrently over its own inventory.

Table 1: End-to-end execution of six open checkpoints through the certified-operator stack: measured offline (dealer-free keygen + certification) and online (compute-only) time, accounted rounds and communication (@512: structural scaling to a full context), LAN/WAN totals using the empirically measured transmission of Table 11, and fidelity against the plaintext float model (boundary in Section 8).

			Time (s)		Rounds		Comm.		Triples	Total (s)		Fidelity	
Model	L	S	offline	online	lin.	FSS	(MB)	@512 (GB)	($\times 10^6$)	LAN	WAN	Cos.	Top-1 (%)
BERT-Base	12	128	18.7	6.6	360	325	812	1.4	312.6	6.9	73.3	0.9663	n/a
GPT-2	12	64	18.4	5.5	336	325	732	1.4	80.8	5.8	67.8	0.9929	96.9
OPT-125M	12	64	18.1	5.5	360	325	741	1.4	80.8	5.8	69.4	0.9697	95.3
Qwen2.5-0.5B	24	64	19.9	23.2	840	745	3 033	4.8	195.6	24.4	223.8	0.9994	98.4
Qwen2.5-1.5B	28	64	21.0	63.1	868	757	10 819	13.8	393.6	66.9	577.1	0.9998	98.4
Llama-3.2-1B	16	64	21.6	71.1	1136	1073	8 020	10.6	292.6	73.9	502.3	0.9961	100.0

A producer–consumer inventory. The scheduler treats certified key material as inventory generated ahead of demand. A *job* is a typed request from the public plan π (§4), namely a correlation type, DPF/DCF key shape, or triple batch, and an *inventory item* is the material satisfying one job with its lifecycle metadata. Each item moves through $\mathsf{planned} \to \mathsf{produced} \to \mathsf{certified} \to \mathsf{ready} \to \mathsf{consumed}$, or to $\mathsf{rejected}$ if its verdict is not accept. Ready items of type τ sit in a FIFO queue Q_τ, each with a fresh nonce η namespaced under the session identifier sid. The producer refills under integer watermarks $\ell < h$: it plans a batch when $|Q_\tau| < \ell$, sizes it by a prefetch window W, and blocks (backpressure) at $|Q_\tau| \geq h$. Every quantity it reads (queue depths, demand vector $\mathbf{d}$, watermarks) is *public*, so scheduling reveals nothing about the inputs. The consumer draws items across the certification barrier of the session state machine (§5), asserts the nonce unspent, marks it consumed, and evaluates the gate; release occurs only at the release barrier once every consumed nonce carries an accept verdict. State is journaled under sid: on a crash the scheduler replays the journal, discards uncertified in-flight batches, and never re-issues a consumed nonce, so recovery cannot resurrect a spent correlation. Algorithm 2 states the two roles, which P_0 and P_1 each run; the $\mathsf{GenTrace}/\mathsf{GenCheck}$ messages within a batch are the only cross-party traffic.

The silent-VOLE instantiation of $\mathcal{F}_{\mathrm{prep}}$, Algorithm 2 with its ready-queue invariants (Proposition 1), the latency decomposition, the Half-Tree backend and the GPU kernels are detailed in Appendix G; their measurements are in §8.2 and Appendix K.

7 Security Analysis

Main theorem. The runtime (Algorithm 1) binds $\mathsf{GenTrace}$, $\mathsf{GenCheck}$, Eval and $\mathsf{Release}$ into one session so that no uncertified key is consumed and no unauthenticated share released. Intuitively, every value the corrupted party sees online is either a uniformly masked opening or an authenticated share, so its whole view is simulatable; every deviation must survive either the certification ledger or the aggregated Λ-MAC, and each is caught except with the core's soundness error. A component-by-component account of the composition is given in Appendix I.

Theorem 3 (Security of Π_{Duplex})**.** *Let $\mathcal{F}_{\mathrm{inf}}$ be the two-party inference functionality for a public architecture M and budget δ. In the $(\mathcal{F}_{\mathrm{prep}}, \mathcal{F}_{\mathrm{coin}})$-hybrid model, Π_{Duplex} (Algorithm 1) securely realises $\mathcal{F}_{\mathrm{inf}}$ with abort against one statically corrupted, rushing, malicious party, under the verification core's prescribed predicate-based abort convention. For m certified instances the distinguishing advantage is at most $m\,\varepsilon_{\mathrm{gen}} + \varepsilon_{\mathrm{rep}} + \mathsf{negl}(\lambda)$, where $\varepsilon_{\mathrm{gen}} \approx 2n/|E|$ and $\varepsilon_{\mathrm{rep}} \leq 2/|E|$ are the core's certification and release soundness (Xue et al., 2026).*

The proof, the arguments that the public plan (C2) and the scheduler (C3) add no secret-dependent behaviour, and the additional assumptions and trust boundaries are given in Appendices H and I.

8 Implementation and Evaluation

Methodology. All experiments run on the platform described in Appendix L (two NVIDIA A100 SXM4 80 GB GPUs, dual-socket Intel Xeon Platinum 83xx CPUs, Python $\geq$3.9, Py-

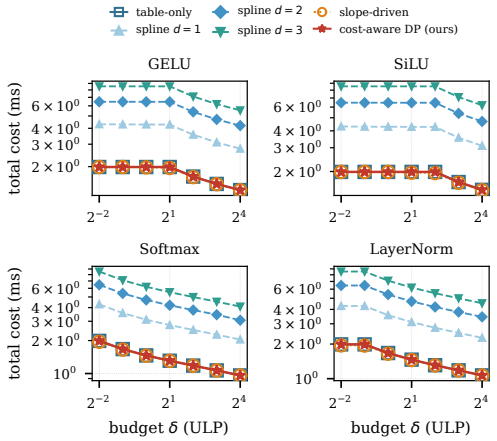


Figure 2: Total cost against the accuracy budget δ for the four mandatory operators: fixed spline degrees, table-only, slope-driven, and the cost-aware DP. The DP tracks the best single mode where one exists and undercuts every fixed-degree spline by 2–4× under active security.

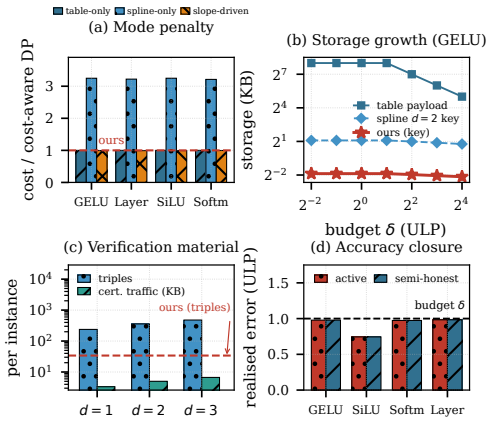


Figure 3: Why the cost-aware plan wins: (a) baseline mode penalty; (b) storage doubles per precision bit for a pure table while the DP plan stays flat; (c) per-instance triples and certification traffic grow with spline degree; (d) realised compiled-gate error stays within the budget.

Torch 2.0.1/CUDA 11.8, EzPC, and OpenSSL AES-NI); fixed point is $Q21.16$; fidelity is against the plaintext float model. Five research questions organise the evaluation. Implementation notes, RQ1 (pipeline correctness and deviation detection; Table 5), RQ4 (the Half-Tree, CUDA and CUTLASS components) and Figure 7 are reported in Appendix K.

What is executed, and its boundary. The C1 protocol (masked-opening handoff, certified gates, session barriers, and the aggregated Λ-MAC release with abort) runs as a two-role protocol on the models of Appendix K.1, where the release check is evaluated and a deviation aborts. The six large language models (Section 8.3) run the same certified-operator stack and exact fixed-point ring GEMM through a real-world two-party network interaction workflow, embedding through output head: a genuine reduced-scale execution, not a composition of per-operator costs. Communication is *accounted* per linear layer (Beaver openings plus the second plane, including the attention products), latency includes network transmission, and the norm reductions use fixed point with the certified truncation gate.

8.1 RQ2: Does cost-aware segmentation reduce active-verification cost?

For GELU, Softmax, SiLU and LayerNorm we solve C2's cost-aware plan under the active profile and price every baseline (table-only, spline-only, fixed-degree, slope-driven, semi-honest-cost) under the same profile. Figure 2 sweeps the budget: at every δ the DP tracks the best single mode and undercuts every fixed-degree spline by 2–4×. Figure 3 decomposes why: the mode penalty (a); table storage doubling per precision bit while the DP plan stays flat (b); triples and certification traffic growing with spline degree (c); and realised compiled error within one ULP (d). Two effects must be kept separate. *Mode awareness*: a spline-only approximant costs up to 3.2× more, because under active security each spline degree carries a certified DCF plane and an authenticated multiplication a table avoids; this 3.2× is the value of choosing the mode under an active cost model. *The semi-honest/active gap itself is modest*: the optimal plans coincide in all but a few cases (a degree-3→2 shift on one non-mandatory operator), so we claim no large security-model-induced divergence; the contribution is a compiler that *prices* verification and provably attains the grid optimum. The plan is a public function of the architecture and δ, so it leaks nothing.

8.2 RQ3: How much preprocessing does producer–consumer overlap hide?

We sweep three load regimes and queue depths from 1 to 16 (Figure 8, Appendix L). Under balanced load the overlap hides preprocessing to a median $1.46\times$ over serial execution; serial and overlapped runs execute the same work, none moved outside the timing window. The six queue invariants (Proposition 1: no uncertified key in the ready queue, one-time consumption, public-only refill, release never crossing the barrier, session isolation, and a transcript unchanged by role overlap) hold in every run.

8.3 RQ5: Does the system run end to end on real open LLMs?

Table 1 reports the execution of six open checkpoints (BERT-Base, GPT-2, OPT-125M, Qwen2.5-0.5B, Qwen2.5-1.5B and Llama-3.2-1B, spanning encoder, decoder, GQA, GELU/ReLU/SiLU activations and LayerNorm/RMSNorm) through the certified-operator secure stack: token embedding, every linear and nonlinear layer, dealer-free key production and certification, and online execution under the boundary stated above. The secure fixed-point output tracks the plaintext model to between 0.9663 and 0.9998 per-token cosine, and where a task head is available the next-token top-1 prediction agrees on 95.3–100% of positions on the fixed evaluation input (Figure 7). Fidelity here is the fixed-point approximation's, not a task-accuracy claim over a dataset, and BERT carries no task head so only its cosine is reported. One further checkpoint, TinyLlama-1.1B, is intrinsically perturbation-sensitive (its own float forward drops to 0.74 cosine under 10^{-4} input noise), so no bounded-precision execution tracks it; it is reported in Appendix L. Communication is large (hundreds of megabytes to gigabytes) because every counted layer's Beaver openings and the second plane are priced at full precision: the cost of dealer-free active security, presented alongside fidelity rather than hidden. The large-language-model runs establish that the stack executes at these scales with usable fidelity, tempered by the reduced sequence lengths and the accounting boundary above.

8.4 Baselines

We organise baselines by a five-tier security-target taxonomy (trusted-dealer + semi-honest, dealer-free + semi-honest, dealer-free + client-malicious, dealer-free + active as in Duplex, and generic active 2PC), because a speedup across incompatible threat models is not meaningful. Table 12 (appendix) compares one certified activation across the tiers. Where the advantage is: the certified gate moves **zero online bytes in zero rounds**. Against the dealer-based BGI DPF lookup executed on the same host and the same expander it answers in 0.14 ms where the full-domain lookup takes 17.0 ms ($118\times$), the $\Theta(2^n)$ expansion having moved into certified preprocessing; against the executed SPDZ-style active 2PC (8.3 KB over 130 rounds per call) the WAN gap is $45\,190\times$; against authenticated garbling (2 KB per gate online) the online cost is likewise nil. Published systems, re-evaluated for each call at their reported operating points, all pay online rounds (7 for SIGMA's GELU, ≈ 4 for the client-malicious GC systems, ≈ 10 for BumbleBee), so one WAN round trip prices an isolated call at 0.20–0.50 s; these cross-tier factors are context, not like-for-like claims. MUSE and SIMC protect against a malicious client only; Duplex tolerates either party. At model scale the same table shows dealer-free active inference at 33–$152\times$ lower WAN latency than composing the same checkpoints on the generic active-2PC row, and in Table 11 (appendix) Duplex has the lowest totals of any dealer-free system on every profile. The disadvantage is stated once: a trusted-dealer GPU system (SIGMA) remains $\approx 6\times$ faster on a LAN (1.2 s vs. 6.9 s on BERT-Base), and dealer-freeness is paid offline (Section 8.2). Duplex is, to our knowledge, the only executed system that is simultaneously dealer-free, purely two-party and actively secure on an end-to-end LLM.

9 Conclusion

We presented Duplex, a two-party runtime that extends the verification core into a dealer-free, actively secure private-LLM-inference system: a session state machine making uncertified consumption or unauthenticated release structurally impossible (C1), a compiler that prices verification into the approximation plan (C2), and a PCG/VOLE scheduler with Half-Tree and bit-exact CUDA/CUTLASS backends (C3). Six checkpoints execute end to end at up to 0.9998 cosine (C4); narrowing the communication cost is the next step.

AI USE STATEMENT

In this work, we used generative AI tools (DeepSeek-V4.1-Flash from DeepSeek Inc.) to help implement and refactor parts of the measurement harness and baseline experiments, perform syntax checks, and assist in cross-checking the manuscript and artifacts. We have not used generative AI tools to generate synthetic data sets, to propose or refine hypotheses, or to interpret results; translation, cleaning or reformatting of data sets, and qualitative or thematic data analysis are not applicable to this work. We have reviewed all AI-assisted work: AI-assisted code is exercised by the 18-check correctness suite (Table 7) and was verified by the authors; every reported number is regenerated from the raw measurement logs (Appendix L); all definitions, theorems and proofs were checked by the authors. We take responsibility for the final content of all materials produced with the aid of generative AI, including code, documentation and artifacts.

ETHICS STATEMENT

This work develops cryptographic protocols whose purpose is to protect the confidentiality of a user's prompt and of a provider's model weights during inference. It involves no human subjects and no personal or proprietary data: the evaluation uses publicly released model checkpoints and a fixed public prompt set (Appendix L). The guarantees are those of the stated threat model (Section 2): DUPLEX protects the protocol transcript, not what a correctly released output may reveal, so defences against model extraction or membership inference remain the deployer's responsibility, and side channels are out of scope (Appendix B). We are not aware of a conflict of interest or of a harmful application specific to this work beyond those of private inference in general.

REPRODUCIBILITY STATEMENT

The complete security proof of Theorem 3 and the optimality proof of Theorem 2, with every assumption they use, are in Appendix I; the operator reductions and measured approximation errors behind the compiler are in Appendix J. The artifact appendix (Appendix L) lists the hardware and software dependencies, the pinned public checkpoints, the four experiments E1–E4 with their expected outputs and tolerances (Table 6), the 18-check correctness suite (Table 7), and the model configurations (Table 8). Every figure, generated table and headline number in the paper is derived from the raw measurement logs by the scripts in the artifact.

REFERENCES

Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 291–308. ACM, 2019a. doi: 10.1145/3319535.3354255.

Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *Advances in Cryptology – CRYPTO 2019*, volume 11694 of *Lecture Notes in Computer Science*, pp. 489–518. Springer, 2019b. doi: 10.1007/978-3-030-26954-8_16.

Nishanth Chandran, Divya Gupta, Sai Lakshmi Bhavana Obbattu, and Akash Shah. SIMC: ML inference secure against malicious clients at semi-honest cost. In *31st USENIX Security Symposium*, pp. 1361–1378. USENIX Association, 2022.

Ke Cheng, Yuheng Xia, Anxiao Song, Jiaxuan Fu, Wenjie Qu, Yulong Shen, and Jiaheng Zhang. Mosformer: Maliciously secure three-party inference framework for large transformers. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2025.

Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In *Advances in Cryptology – CRYPTO 2018*, volume 10992 of *Lecture Notes in Computer Science*, pp. 769–798. Springer, 2018. doi: 10.1007/978-3-319-96881-0_26.

Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pp. 643–662. Springer, 2012. doi: 10.1007/978-3-642-32009-5_38.

Jack Doerner and Abhi Shelat. Scaling ORAM for secure computation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 523–535. ACM, 2017. doi: 10.1145/3133956.3133967.

Caiqin Dong, Jian Weng, Jia-Nan Liu, Yue Zhang, Yao Tong, Anjia Yang, Yudan Cheng, and Shun Hu. Fusion: Efficient and secure inference resilient to malicious servers. In *30th Annual Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2023a.

Ye Dong, Wen-jie Lu, Yancheng Zheng, Haoqi Wu, Derun Zhao, Jin Tan, Zhicong Huang, Cheng Hong, Tao Wei, and Wenguang Chen. PUMA: Secure inference of LLaMA-7B in five minutes. arXiv:2307.12533, 2023b.

Xiaojie Guo, Kang Yang, Xiao Wang, Wenhao Zhang, Xiang Xie, Jiang Zhang, and Zheli Liu. Half-Tree: Halving the cost of tree expansion in COT and DPF. In *Advances in Cryptology – EUROCRYPT 2023*, volume 14004 of *Lecture Notes in Computer Science*, pp. 330–362. Springer, 2023. doi: 10.1007/978-3-031-30545-0_12.

Kanav Gupta, Deepak Kumaraswamy, Nishanth Chandran, and Divya Gupta. LLAMA: A low latency math library for secure inference. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2022(4):274–294, 2022. doi: 10.56553/popets-2022-0109.

Kanav Gupta, Neha Jawalkar, Ananta Mukherjee, Nishanth Chandran, Divya Gupta, Ashish Panwar, and Rahul Sharma. SIGMA: Secure GPT inference with function secret sharing. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2024(4):61–79, 2024. doi: 10.56553/popets-2024-0107.

Kanav Gupta, Nishanth Chandran, Divya Gupta, Jonathan Katz, and Rahul Sharma. SHARK: Actively secure inference using function secret sharing. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 2472–2490, 2025. doi: 10.1109/SP61157.2025.00175. Full version: IACR ePrint 2025/716.

Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. Iron: Private inference on transformers. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, pp. 15718–15731, 2022.

Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. Cheetah: Lean and fast secure two-party deep neural network inference. In *31st USENIX Security Symposium*, pp. 809–826. USENIX Association, 2022.

Neha Jawalkar, Kanav Gupta, Arkaprava Basu, Nishanth Chandran, Divya Gupta, and Rahul Sharma. Orca: FSS-based secure training and inference with GPUs. In *2024 IEEE Symposium on Security and Privacy (S&P)*, pp. 597–616. IEEE, 2024. doi: 10.1109/SP54263.2024.00063.

Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium*, pp. 1651–1669. USENIX Association, 2018.

Ryan Lehmkuhl, Pratyush Mishra, Akshayaram Srinivasan, and Raluca Ada Popa. Muse: Secure inference resilient to malicious clients. In *30th USENIX Security Symposium*, pp. 2201–2218. USENIX Association, 2021.

Dacheng Li, Hongyi Wang, Rulin Shao, Han Guo, Eric P. Xing, and Hao Zhang. MPCFormer: Fast, performant and private transformer inference with MPC. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Zhengyi Li, Kang Yang, Jin Tan, Wen-jie Lu, Haoqi Wu, Xiao Wang, Yu Yu, Derun Zhao, Yancheng Zheng, Minyi Guo, and Jingwen Leng. Nimbus: Secure and efficient two-party inference for transformers. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024.

Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and WenGuang Chen. BumbleBee: Secure two-party inference framework for large transformers. In *32nd Annual Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2025.

Yunlv Lv, Rui Zhang, Zhiyuan Zhang, Ziyi Wan, Lanxue Zhang, Minhui Xue, Jiangtao Li, and Yanan Cao. SMASH: Scalable maliciously secure hybrid multi-party computation framework for privacy-preserving large language models. In *35th USENIX Security Symposium*. USENIX Association, 2026. URL https://www.usenix.org/conference/usenixsecurity26/presentation/lv.

Yuhan Ma, Yong Li, and Stefan Schmid. FuseFSS: Efficient secure LLM inference with function secret sharing. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. Preprint: arXiv:2606.09551.

Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. Delphi: A cryptographic inference service for neural networks. In *29th USENIX Security Symposium*, pp. 2505–2522. USENIX Association, 2020.

Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. BOLT: Privacy-preserving, accurate and efficient inference for transformers. In *2024 IEEE Symposium on Security and Privacy (S&P)*, pp. 4753–4771. IEEE, 2024. doi: 10.1109/SP54263.2024.00130.

Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. CrypTFlow2: Practical 2-party secure inference. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 325–342. ACM, 2020. doi: 10.1145/3372297.3417274.

Théo Ryffel, Pierre Tholoniat, David Pointcheval, and Francis Bach. AriaNN: Low-interaction privacy-preserving deep learning via function secret sharing. *Proceedings on Privacy Enhancing Technologies*, 2022(1):291–316, 2022. doi: 10.2478/popets-2022-0015.

Kyle Storrier, Adithya Vadapalli, Allan Lyons, and Ryan Henry. Grotto: Screaming fast $(2+1)$-PC for $\mathbb{Z}_{2^n}$ via $(2,2)$-DPFs. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 2143–2157. ACM, 2023. doi: 10.1145/3576915.3623147.

Zhengyuan Su, Qi Pang, Simon Beyzerov, and Wenting Zheng. FABLE: Batched evaluation on confidential lookup tables in 2PC. In *34th USENIX Security Symposium*, pp. 1945–1964. USENIX Association, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/su-zhengyuan.

Sameer Wagh. Pika: Secure computation using function secret sharing over rings. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2022(4):351–377, 2022. doi: 10.56553/popets-2022-0113.

Yujie Xue, Lin Liu, Rongmao Chen, Yizhen Jiang, Yuchuan Luo, Bing Sun, and Shaojing Fu. Actively secure two-party function secret sharing with dynamic cross-phase verification. Cryptology ePrint Archive, Paper 2026/1697, 2026. URL https://eprint.iacr.org/2026/1697.

Chenkai Zeng, Qi Feng, Debiao He, and Min Luo. LightShark: Actively secure machine-learning inference based on lightweight authenticated distributed comparison function. *Cryptology ePrint Archive, Paper 2026/1487*, 2026. URL https://eprint.iacr.org/2026/1487.

APPENDIX OVERVIEW

Appendix A surveys related work and Appendix B discusses limitations. Appendices C–H and Appendix K reproduce verbatim, in their original order, the passages deferred from the corresponding sections of the main text; in them, "this section" refers to the main-text section together with its extension. Likewise, a reference in these passages to a numbered main-text section includes that section's extension; in particular, statements said to be proved in Section 7 are proved in Appendices H and I.1, with the queue invariants in Proposition 1. Appendix I contains the deferred proofs, Appendix J the operator compilation details, and Appendix L the artifact appendix with the extended results.

A RELATED WORK

For every technique DUPLEX reuses we mark the security target it was designed for, so that what is inherited stays separable from what is added.

FSS-based private inference. FSS is the most effective route for the nonlinear layers because each gate moves only a few masked-opening bytes once keys are placed, never function-dependent traffic: SIGMA (Gupta et al., 2024) evaluates a GPT-scale Transformer this way and LLAMA (Gupta et al., 2022) supplies the FSS math gates. Both assume a *trusted dealer* and *semi-honest* evaluators; DUPLEX keeps their gate-level online structure but targets a setting where neither holds.

Dealer-free active FSS. The verification core (Xue et al., 2026) removes the dealer and the semi-honest restriction together: dealer-free generation secure against one static malicious party, a two-plane authenticated key (Δ offline, Λ FSS outputs), a certification chain, and a lifting theorem giving active security at no online cost to any affine post-processing of a constant number of DPF/DCF evaluations. DUPLEX inherits its protocols, theorems, security model and costs unchanged and builds an inference system around it.

Table and spline approximation. Pika (Wagh, 2022) evaluates a *public* table at a private index (dealer-assisted); Grotto (Storrier et al., 2023) realises splines in the $(2{+}1)$-party setting from a single DPF; FABLE (Su et al., 2025) does batched two-party lookups over a *server-secret* table (a distinct setting); Floram (Doerner & Shelat, 2017) originates full-domain FSS-for-read/write for ORAM. Each fixes a single mode or party model under semi-honest or dealer-assisted security; DUPLEX instead prices tables and low-degree splines jointly under dealer-free active security and solves for the per-region plan (Section 4).

Preprocessing, compression, and GPU. PCGs (Boyle et al., 2019b) and silent VOLE (Boyle et al., 2019a) expand short seeds under dual-LPN; DUPLEX instantiates $\mathcal{F}_{\mathrm{prep}}$ from silent subfield-VOLE and schedules its production against online consumption (Section 6). Half-Tree (Guo et al., 2023) halves per-level seed corrections under a circular correlation-robust hash, adopted here as the key backend at that one extra assumption. Orca (Jawalkar et al., 2024) and SIGMA move FSS onto GPUs in the semi-honest dealer-assisted regime; DUPLEX carries the same target into the active setting with bit-exact CUDA/CUTLASS ring-GEMM kernels.

Two-party transformer baselines. CrypTFlow2 (Rathee et al., 2020), Iron (Hao et al., 2022), BOLT (Pang et al., 2024) and BumbleBee (Lu et al., 2025) combine homomorphic encryption with OT/GC for two-party inference at increasing scale, predominantly semi-honest and with a communication profile distinct from the FSS route. We place these, the FSS systems above, and DUPLEX in a five-tier taxonomy so every comparison is like-for-like (Section 8); no prior FSS inference system operates in the dealer-free, actively secure regime DUPLEX targets.

B DISCUSSION AND LIMITATIONS

Security model and assumptions. DUPLEX inherits the verification core's guarantees (Xue et al., 2026): security *with abort* against one static malicious rushing party. Every deviation is caught

Table 2: Positioning against prior secure-inference systems along the axes that matter for private LLM inference (from the cited descriptions). *Setting*: primitive for the nonlinear layers and party model (+D: a trusted dealer or helper supplies keys/correlated randomness). *Adv.*: SH semi-honest; MC/MS malicious client/server only; M malicious, either party, with abort; M_{hm} malicious with an honest majority (1-of-3); M_{dm} malicious, dishonest majority. *Certified keys*: preprocessing material is verified by the evaluators before use. *Cost-aware*: the nonlinear approximation is chosen by an explicit cost/accuracy optimisation. *Pipelined*: × no input-independent preprocessing phase; ○ an offline phase without overlap; ✓ preprocessing overlapped with online execution by a producer–consumer scheduler. SPDZ-style 2PC: Damgård et al. (2012); Cramer et al. (2018). †end-to-end cost reported for the nonlinear layers.

System	Setting	Adv.	Dealer-free	LLM e2e	GPU	Certified keys	Cost-aware	Pipelined
Trusted dealer or helper, semi-honest								
AriaNN (Ryffel et al., 2022)	FSS, 2PC+D	SH	×	×	✓	×	×	○
LLAMA (Gupta et al., 2022)	FSS, 2PC+D	SH	×	×	×	×	×	○
Grotto (Storrier et al., 2023)	FSS, (2+1)PC	SH	×	×	×	×	×	○
MPCFormer (Li et al., 2023)	SS, 2PC+D	SH	×	✓	✓	×	×	○
SIGMA (Gupta et al., 2024)	FSS, 2PC+D	SH	×	✓	✓	×	×	○
Orca (Jawalkar et al., 2024)	FSS, 2PC+D	SH	×	×	✓	×	×	○
FuseFSS (Ma et al., 2026)	FSS, 2PC+D	SH	×	✓	✓	×	×	○
Dealer-free, semi-honest								
Gazelle (Juvekar et al., 2018)	HE+GC, 2PC	SH	✓	×	×	×	×	○
Delphi (Mishra et al., 2020)	HE+GC, 2PC	SH	✓	×	×	×	✓	○
CrypTFlow2 (Rathee et al., 2020)	OT+HE, 2PC	SH	✓	×	×	×	×	×
Cheetah (Huang et al., 2022)	HE+OT, 2PC	SH	✓	×	×	×	×	×
Iron (Hao et al., 2022)	HE+OT, 2PC	SH	✓	✓	×	×	×	×
BOLT (Pang et al., 2024)	HE+OT, 2PC	SH	✓	✓	×	×	×	×
Nimbus (Li et al., 2024)	HE+OT, 2PC	SH	✓	✓	×	×	×	○
BumbleBee (Lu et al., 2025)	HE+OT, 2PC	SH	✓	✓	×	×	×	×
PUMA (Dong et al., 2023b)	RSS, 3PC	SH	✓	✓	×	×	×	×
Dealer-free, one-sided malicious								
MUSE (Lehmkuhl et al., 2021)	HE+GC, 2PC	MC	✓	×	×	×	×	○
SIMC (Chandran et al., 2022)	HE+GC, 2PC	MC	✓	×	×	×	×	○
Fusion (Dong et al., 2023a)	HE+OT, 2PC	MS	✓	×	×	×	×	×
Malicious, with a trusted dealer or an honest majority								
Pika (Wagh, 2022)	FSS, (2+1)PC	M_{hm}	×	×	×	✓	×	○
SHARK (Gupta et al., 2025)	FSS, 2PC+D	M	×	✓	×	×	×	○
LightShark (Zeng et al., 2026)	FSS, 2PC+D	M	×	✓	×	×	×	○
Mosformer (Cheng et al., 2025)	FSS/RSS, 3PC	M_{hm}	✓	✓	×	✓	×	○
Dealer-free, dishonest majority								
SPDZ-style 2PC	auth. shares, 2PC	M_{dm}	✓	×	×	✓	×	○
SMASH (Lv et al., 2026)	HE+LUT, n-party	M_{dm}	✓	✓†	×	✓	×	○
DUPLEX (this work)	**FSS, 2PC**	**M**	✓	✓	✓	✓	✓	✓

by the core, and DUPLEX collapses its per-wire failure events into a single execution-wide abort: a cheating attempt achieves nothing beyond termination, and the prompt, the weights and every activation remain secret whether the protocol completes or aborts. Guaranteed-output deployments fall outside the model. Offline security reduces to dual-LPN at the fixed VOLE parameters (Boyle et al., 2019a;b); Half-Tree adds a fixed-offset circular correlation-robust hash, no weaker than the fixed-key AES already assumed.

Communication, state, and device. The scheduler hides preprocessing behind compute (balanced overlap near $1.46\times$) but leaves total communication unchanged: full-precision Beaver openings and the tag plane yield 0.7–11 GB transcripts, and bandwidth remains the binding resource. Key storage scales with the number of protected gates; without checkpointing, a crash between CERTIFIED and

RELEASED forces regenerating the consumed correlations. The GPU backend is bit-exact against the CPU reference but is a trusted local device; microarchitectural side channels are out of scope.

Evaluation scope and fidelity. Large-language-model runs use reduced contexts (Llama at 64 tokens) and $Q21.16$ arithmetic, so long-context accuracy is not established here. DUPLEX protects the transcript, not what a released output reveals: model-extraction and membership-inference defenses layer on the output.

C EXTENDED INTRODUCTION AND BACKGROUND

Results in brief. DUPLEX runs end to end on six real checkpoints. On BERT-Base, GPT-2 and Llama-3.2-1B the secure fixed-point output tracks the plaintext model to 0.9663, 0.9929 and 0.9961 per-token cosine, with next-token top-1 agreement of 96.9% (GPT-2) and 100.0% (Llama); the authenticated-release-with-abort property is demonstrated as a two-role protocol (Appendix K.1). A naive spline-only approximant costs up to $3.2\times$ more than the cost-aware plan under active security. The exact secret-shared ring GEMM runs at 57 GMAC/s on the GPU versus 9.8 GMAC/s on the CPU ($5.8\times$, operands device-resident) while matching it bit for bit, and Half-Tree brings seed-correction key material to 0.52 of the baseline.

Scope. Security is with abort against one statically corrupted malicious rushing party in the $\mathcal{F}_{\text{prep}}$-hybrid model. A cheating party gains no advantage beyond the ability to force a premature termination: every deviation is caught by the verification core, and the abort signal itself, inherent to any protocol in the security-with-abort paradigm, discloses no useful information about the parties' secrets. Architecture, tensor shapes, context length, fixed-point format, operator catalogue and schedule are public; the prompt, the weights and every activation remain secret throughout, whether the protocol completes or aborts. We evaluate at reduced sequence lengths, stated with every result. Private LLM inference that resists a *malicious* party must certify every secret-shared computation before any value leaves the protocol, and must do so without a trusted dealer. This section fixes notation, summarizes the two-party function secret sharing (FSS) verification core that DUPLEX builds on, and states the system and threat model under which the DUPLEX runtime operates. The verification core and its security definition are attributed throughout to Xue et al. (2026); we mark precisely what DUPLEX inherits and defer the new components to Sections 5–8.

This section established the notation, the inherited verification-core interface and its lifting guarantee, and the system and threat model under which the DUPLEX runtime and its new components are built and evaluated in the remainder of the paper.

D EXTENDED SYSTEM OVERVIEW

The verification core (Xue et al., 2026) settles the cryptographic question one gate at a time: dealer-free key generation, a single DPF/DCF-derived gate certified against one static malicious party, and a lifting theorem extending that guarantee to any gate built by public affine post-processing of a constant number of function-secret-sharing (FSS) evaluations. But a private LLM inference is not one gate: it is thousands of nonlinear gates interleaved with linear layers, fixed-point truncations, and a single final release, all bound to one session. This raises the first design question of DUPLEX.

Q1: If the verification core already certifies each gate dealer-free and against a malicious party, why is it not yet a deployable end-to-end secure LLM inference framework?

Because per-gate certification is local and an inference needs a global guarantee. The core certifies that a given key realises its gate; it says nothing about *which* keys a session may consume, in *what* order, or when a masked activation may be opened. A malicious party that consumes a key before its verdict is known, or induces a release of an activation whose authentication tag was never checked, defeats the guarantee without breaking a single gate. Turning the core into a framework therefore requires a runtime in which consuming an uncertified key and releasing an unauthenticated output are structurally impossible rather than merely detected. That runtime, with the compiler and scheduler that feed it, is the contribution of DUPLEX; its architecture is shown in Figure 1.

Actors and trust. Two parties P_0 and P_1 jointly evaluate a public model architecture on secret inputs: P_0 holds the prompt, P_1 the weights, and neither learns the other's secret beyond the released output. The model dimensions (S context length, d width, d_{ff} FFN width, H heads, L layers), the operator catalogue, the fixed-point format (Q21.16), and the schedule are public; the prompt, the weights, and every activation are secret. DUPLEX inherits the verification core as a black box (Xue et al., 2026): its two-plane authenticated key $F_{\theta,\Lambda}$, dealer-free distributed generation GenTrace, certification GenCheck, authenticated release Release, and cross-phase verification chain are used unchanged and cited in the third person. The security backbone reuses that chain; the runtime, compiler, and scheduler layered on top are new.

A single layer, end to end. One Transformer layer traverses all four stages. *Compile:* the public layer shape fixes the QKV and FFN GEMM dimensions, the LUT/spline plans for its Softmax, LayerNorm and GELU/SiLU activations, and the inventory of authenticated triples and DPF/DCF keys they consume (Section 4). *Produce:* the background producer draws that inventory from silent subfield-VOLE, derandomises it into authenticated Beaver triples, certifies every key bundle with GenCheck, and enqueues it (Section 6). *Execute:* the consumer opens masked shares against certified triples for the linear sublayers, truncates back to Q21.16, and evaluates each nonlinearity as an offset gate $G^{[r_{\text{in}},r_{\text{out}}]}(\hat{x}) = G(\hat{x} - r_{\text{in}}) + r_{\text{out}}$ on the public masked wire $\hat{x} = x + r_{\text{in}}$ (Section 5); offline wires stay under the Δ-MAC key as $[\![x]\!]$ and FSS outputs under the Λ-MAC key as $\langle\langle x \rangle\rangle$, so no value leaves the authenticated share domain between layers. *Release:* only after the last layer, and only if every certification verdict passed, does Release open the output under the aggregated Λ-MAC (Section 7).

Session state machine. The runtime enforces these rules as a session state machine, making the safety properties structural rather than checked by convention. A session advances through OPEN $\to$ PLANNED $\to$ PRODUCING $\to$ CERTIFIED $\to$ ONLINE $\to$ RELEASED: it plans from public architecture, produces material, crosses the certification barrier into CERTIFIED, executes, and releases. Within a session each unit of key material has its own lifecycle, and a unit may be consumed by the online path only once it is *ready*:

- **not-produced** $\to$ **generated**: GenTrace emits a candidate key bundle, still uncertified.
- **generated** $\to$ **certified**: GenCheck returns a passing verdict; this transition is the *certification barrier*.
- **certified** $\to$ **ready**: the unit enters the authenticated ready queue and becomes eligible for one-time consumption.
- **ready** $\to$ **consumed**: the online consumer uses the unit exactly once; a failing verdict instead sends it to **rejected** and the session aborts without release.

No unit reaches **ready** without passing the barrier, consumption is one-time, and Release fires only from ONLINE, so a key cannot be used before its verdict and an output cannot be released before certification. These invariants are proved in Section 7 and exercised empirically in Appendix K.1, where an injected-deviation campaign confirms that honest runs release and injected malicious deviations abort with no release.

Positioning. To our knowledge, DUPLEX is the only two-party secure-inference system that is at once dealer-free and actively secure while running a modern LLM end to end. Table 2 (Appendix A) places it against prior FSS-, HE/OT- and MPC-based systems along the security and systems axes that matter for private inference: party model and dealer, adversary, LLM-scale execution, GPU support, certified key material, cost-aware approximation and pipelined preprocessing.

Roadmap. The four contributions map onto the technical sections and their evaluation: the secure runtime and session state machine (C1) in Section 5 and Section 7, measured in Appendix K.1; the cost-aware LUT/spline compiler (C2) in Section 4 and Section 8.1; the PCG/VOLE producer–consumer scheduler with its Half-Tree and CUDA/CUTLASS backends (C3) in Section 6 and Section 8.2; and the six-checkpoint end-to-end evaluation (C4) in Section 8.

This section established the architecture of DUPLEX (actors and state, the stages that turn a public model into an authenticated inference, the single-layer path threading them, and the session state

machine whose certification barrier and one-time consumption make consuming an uncertified key or releasing an unauthenticated output structurally impossible), which the sections that follow instantiate and prove secure under one static malicious corruption.

E Additional Compiler Details

Q2: Once verification is priced, why are the slope-driven heuristics and fixed segmentation of the semi-honest setting no longer optimal, and how do the curvature of an operator and the overhead of active verification jointly determine where a region should be a table and where it should be a spline?

The following sketch accompanies Theorem 2.

Proof sketch. Feasible regions are the edges of a DAG on states (i, s); every δ-feasible grid plan is a path $(0, \varnothing) \to (N, \cdot)$ of equal weight, since bundles are charged once and all else is additive, so Bellman optimality gives the minimum-cost path. The bounds follow from $4(N{+}1)$ states and $O(N)$ endpoints across $D{+}1$ modes (Appendix I). □

Compile–measure–tighten. The oracle uses the analytic surrogates $\frac{1}{2}|f'|h$ and $\frac{1}{8}|f''|h^2$, which ignore rounding into the ring and truncation inside the offset gate. To bind the *realised* error the compiler closes a loop: it compiles the emitted plan to concrete Q21.16 offset-gate parameters, measures the realised error $e^\star$ of the actual gate outputs (quantisation, truncation and both authentication planes included) and, if $e^\star > \delta$, re-solves with $\delta \leftarrow \delta \cdot (\delta/e^\star)$. Tightening only removes feasible regions and the realised error exceeds the surrogate by at most a fixed quantisation slack, so the sequence is monotone and terminates within δ; GELU, Softmax, SiLU and LayerNorm each close within one output ULP (Appendix J).

Range reduction. The solver assumes a well-conditioned kernel on a narrow interval, so each operator is reduced first: rsqrt is fitted on a single binade (LayerNorm's and RMSNorm's normalisation reuse it), GELU is clamped to a finite band with offset-gate tails, SiLU is folded onto $x \geq 0$ by $\sigma(x) = 1 - \sigma(-x)$, and Softmax subtracts the row maximum before a bounded e^u and a reciprocal. GELU, Softmax, SiLU and LayerNorm are the mandatory catalogue; the per-operator reductions and the comparison with prior table and spline gates are in Appendix J.

Section 8.1 measures the consequence. A spline-only approximant costs up to $3.2\times$ more than the cost-aware plan under active security, and pricing verification shifts the optimal spline degree *down* (for example from degree three to degree two on an inflected operator), so segmentation, not degree, is the lever once verification is priced. The semi-honest/active divergence is modest but systematic: active security buys segments and tables where the semi-honest optimum buys degree. Running once per operator at compile time on the public architecture and budget, the plan is a public function of the architecture and δ and leaks nothing about the secret input.

This section established a plan representation (Definition 1), a cost model under which table and spline gates are affine and add in the Λ plane so a per-region hybrid needs no secure selection, and a grid-optimal dynamic program (Theorem 2) with a compile–measure–tighten loop certifying realised error within δ: the priced compiler whose plans the scheduler of Section 6 produces and the runtime of Section 3 consumes.

F Additional Runtime Details

A certified FSS key is useful only if it is later evaluated on the value the parties agreed to compute; a runtime that lets a party certify one bundle and consume another, evaluate a gate before its verdict is known, or reconstruct an output before the final check inherits none of the core's guarantees. This section describes the Duplex runtime, whose contribution (C1) is a session state machine and a transcript binding that chain dealer-free generation, certification, online evaluation, and release into one authenticated object. The generation primitive GenTrace, the certification predicate GenCheck with its six-residual moment ledger, the aggregated opening Release, and the cross-domain Aggregate are inherited from the verification core (Xue et al., 2026); new here is the mechanism that forces each to refer to the same planned computation on the same certified keys.

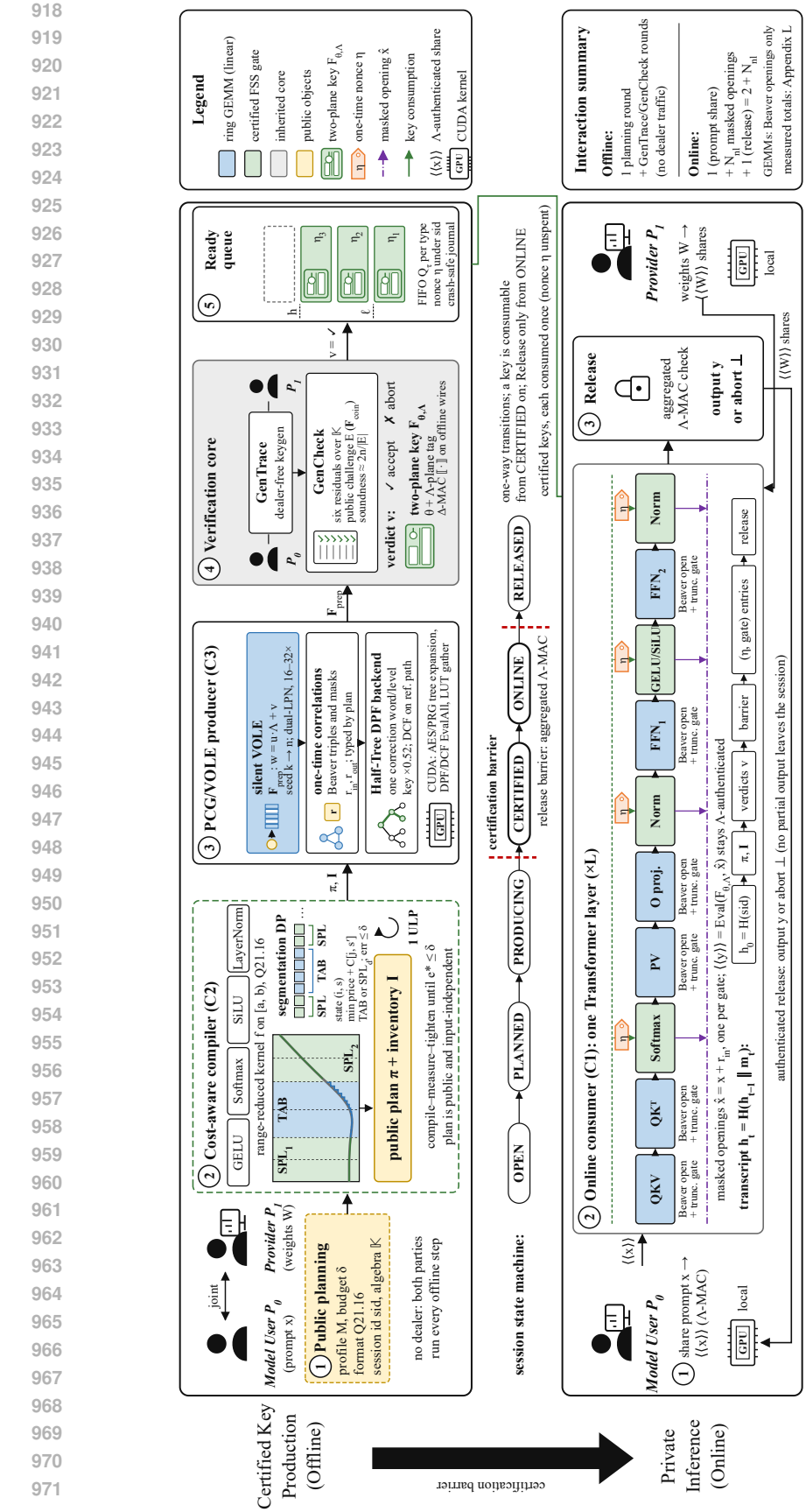


Figure 4: Detailed version of Figure 1 (DUPLEX workflow), set sideways at full page height. It is the same drawing with the per-module sub-annotations that the main-text copy omits for legibility: the DP state and cost recursion of the compiler; the LPN expansion ratio, the typed masks, the key-size ratio and the DCF path of the producer, with the AES/PRG expansion named among its CUDA kernels; the coin-flipping challenge source and soundness bound of GenCheck and the Δ-MAC plane of the key; the queue journal; the per-GEMM Beaver openings and truncation gates of the online layer; and the GEMM line of the interaction summary. The stages are described in the caption of Figure 1.

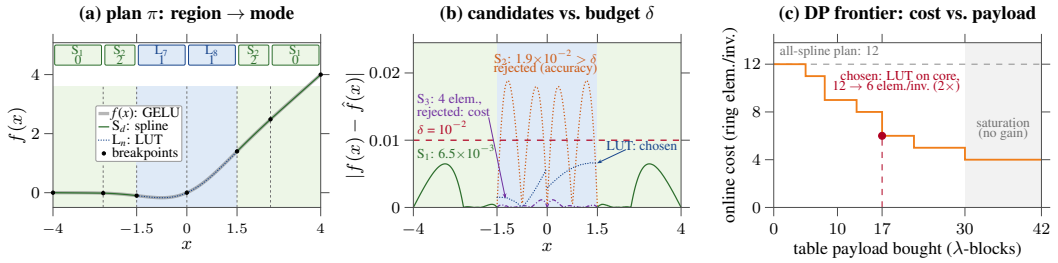


Figure 5: **Cost-aware nonlinear compilation**, worked on GELU over $[-4, 4]$ at an illustrative $\delta = 10^{-2}$ in output units (the deployed budget is one ULP). (a) One mode per region (spline $\mathrm{S}_d \equiv \mathrm{SPL}_d$ on the smooth tails, table $\mathrm{L}_n \equiv \mathrm{TAB}$ with 2^n cells on the curved core) gives the plan π; the chips give each region's mode and, below it, the ring elements it opens per invocation. (b) On the core a degree-2 spline breaks δ and a degree-3 spline meets it only at four opened elements, so tables win. (c) Re-solving as the price of table payload varies traces the cheapest δ-feasible plans: the frontier steps down as more payload is bought, the cheapest swaps first, and saturates beyond 30 blocks (grey); the marked point tables the core only and halves the online cost, counted in ring elements opened per invocation ($2(d-1)$ for a degree-d spline's Horner recombination, one for a table).

Input: grid $x_0 < \ldots < x_N$, budget δ, degree cap D, oracles $\mathrm{err}(\cdot)$, $\mathrm{price}(\cdot)$.
Output: minimum-cost δ-feasible plan.

for $s \in \{0,1,2,3\}$: $C[N, s] \leftarrow 0$
for $i \leftarrow N-1$ **downto** 0:
  for $s \in \{0,1,2,3\}$: $best \leftarrow \infty$
    for $j \leftarrow i+1$ **to** N:
      for $m \in \{\mathrm{TAB}\} \cup \{\mathrm{SPL}_d\}_{d=1}^{D}$:
        if $\mathrm{err}(i, j, m) \le \delta$:
          $(\Delta c,\ s') \leftarrow \mathrm{price}(i, j, m, s)$
          if $\Delta c + C[j, s'] < best$:
            $best \leftarrow \Delta c + C[j, s']$; $\mathrm{nxt}[i, s] \leftarrow (j, m)$
    $C[i, s] \leftarrow best$
return $\textsc{Trace}\big(\mathrm{nxt}, (0, \varnothing)\big)$

Figure 6: **Segmentation solver.** A backward dynamic program over grid atoms and the two-bit fixed-bundle state. Each transition extends the current region to x_j as a table (TAB) or degree-d spline (SPL_d), pays the marginal cost plus any unpaid fixed bundle via price, and keeps the region only if its realised error clears δ. Runs once per operator at compile time.

Stages and their interfaces. Execution proceeds in four stages under a fixed session identifier sid, each named by the artifacts it reads and writes. *Generation* runs GenTrace in the $(\mathcal{F}_{\mathrm{prep}}, \mathcal{F}_{\mathrm{coin}})$-hybrid: from the execution plan π (the ordered grid atoms, offset gates, and linear shapes emitted by the compiler of §4) and correlated randomness it writes a two-plane authenticated key $F_{\theta,\Lambda}$ per gate and an inventory $\mathcal{I}$ of authenticated triples and mask pairs. *Certification* runs GenCheck over $\mathcal{I}$ and emits a per-bundle accept/reject *verdict* v aggregated across the session; a rejecting verdict aborts. *Evaluation* consumes each certified key once to transform Λ-authenticated activation shares layer by layer. *Release* runs the aggregated Λ-MAC check and reconstructs the output only if it passes. Each stage advances the session state and absorbs its outputs into the transcript before the next may read them.

Masked-opening handoff. Activations pass between layers by masked opening. For a nonlinear gate with certified masks $r_{\mathrm{in}}, r_{\mathrm{out}}$, the parties open $\hat{x} = x + r_{\mathrm{in}}$, uniform over the ring and hence hiding x, then evaluate the certified offset gate $G^{[r_{\mathrm{in}}, r_{\mathrm{out}}]}(\hat{x}) = G(\hat{x} - r_{\mathrm{in}}) + r_{\mathrm{out}}$ on the public $\hat{x}$ via Eval, obtaining a fresh Λ-authenticated sharing $\langle\langle y \rangle\rangle$ whose masks come from the same certified key. Only the masked value $\hat{x}$ crosses the wire, and the gate is the one already fixed by the transcript, so x can affect the execution only through the previously agreed abort condition.

Algorithm 1 Π_{Duplex}: dealer-free actively secure inference (C1)

Require: public architecture M, budget δ; P_0 holds x, P_1 holds W

Plan (state OPEN$\to$PLANNED):

1: both parties compute the plan $\pi \leftarrow \mathsf{Compile}(M, \delta)$ and the inventory $\mathcal{I}$; absorb $(\mathsf{sid}, \pi, \mathcal{I})$ into the transcript hash h

Produce ($\to$ PRODUCING), background producer:

2: **for** each job $j \in \mathcal{I}$ **(batched) do**

3: draw correlations from $\mathcal{F}_{\mathrm{prep}}$ (silent-VOLE expansion)

4: $(k_b^{(j)}, tr_j) \leftarrow \mathsf{GenTrace}([\![\theta_j]\!])$ ▷ dealer-free, interactive

5: **end for**

Certify ($\to$ CERTIFIED), the certification barrier:

6: $v \leftarrow \mathsf{GenCheck}(\{tr_j\})$ under one challenge (Aggregate); absorb v into h; **if** $v \neq$ accept **then abort**

7: enqueue only accepted keys, each with a one-time nonce

Execute ($\to$ ONLINE), online consumer:

8: **for** each layer ℓ of M **do**

9: linear: authenticated ring GEMM + certified truncation

10: nonlinear: assert verdict and nonce unspent; open $\hat{x} = x + r_{\mathrm{in}}$; $\langle\!\langle y \rangle\!\rangle \leftarrow \mathsf{Eval}(F_{\theta,\Lambda}, \hat{x})$; mark consumed

11: **end for**

Release ($\to$ RELEASED):

12: $y \leftarrow \mathsf{Release}(\langle\!\langle x_L \rangle\!\rangle)$: aggregated Λ-MAC check; **if** it fails **then abort** (no output)

Preserving the authentication invariant. Every online value carries $h = \Lambda \cdot x$: the parties hold additive shares of x and of its Λ-MAC h. Linear layers preserve it because authenticated Beaver multiplication and the exact limb-decomposed ring GEMM act Λ-linearly on the shares, and truncation re-establishes it as a certified offset gate rather than an unauthenticated shift. Nonlinear layers preserve it because the handoff discards the incoming $\langle\!\langle x \rangle\!\rangle$ only after producing $\langle\!\langle y \rangle\!\rangle$ directly from certified key material, so the output is authenticated under the same Λ. Offline wires authenticated under the second key, $[\![\cdot]\!]$ with MAC key Δ, are folded in through Aggregate so cross-domain values enter the online plane already carrying $h = \Lambda \cdot x$. Composition therefore never leaves a value unauthenticated between a linear and an FSS layer.

The release barrier. Release is permitted only from ONLINE and only after Release passes the aggregated Λ-MAC check over all outputs certified in the session. The single aggregated opening amortizes the check across the transcript, and because the release event is itself absorbed into h_t, a party cannot reconstruct a partial output and then disavow the session. If the check fails the runtime aborts and emits nothing, so a deviation that survives certification is still caught at this final barrier.

Security guarantee. The end-to-end guarantee is stated informally here and proved in §7.

Theorem 4 (Secure inference with abort; informal)**.** *Against a single static malicious party with rushing, in the $(\mathcal{F}_{\mathrm{prep}}, \mathcal{F}_{\mathrm{coin}})$-hybrid, the* DUPLEX *runtime realizes private LLM inference with abort: any accepting transcript releases only the correct output of the planned computation; every deviation is detected before the release barrier except with probability negligible in the statistical security parameter; and the adversary's remaining influence is confined to the previously agreed abort condition.*

The soundness of certification is $\approx 2n/|E|$ over the audit algebra $\mathbb{K}$ ($E \subseteq \mathbb{K}$ the public challenge set), and the transcript binding reduces any cross-stage substitution to a hash collision or a missing verdict. The runtime runs end to end on real checkpoints: an honest run certifies and releases, an injected malicious deviation on any authenticated wire drives the session to ABORT with no release, and any attempt to consume a key before its accept verdict is refused at the certification barrier.

This establishes that generation, certification, evaluation, and release form one binding chain: a certified key is consumable only on the value the transcript fixed, the invariant $h = \Lambda \cdot x$ survives across linear and FSS layers, and a result leaves only after the aggregated Λ-MAC check, leaving the cost of these gates to the compiler of §4.

G Additional Preprocessing and Hardware Details

Instantiating $\mathcal{F}_{\text{prep}}$ with silent subfield-VOLE. The verification core is stated in the $(\mathcal{F}_{\text{prep}}, \mathcal{F}_{\text{coin}})$-hybrid, leaving $\mathcal{F}_{\text{prep}}$ abstract; we realise its bulk with a silent subfield VOLE (Boyle et al., 2019a;b) over the audit algebra $\mathbb{K}$ with subfield $\mathbb{F}_p$ ($p = 2^{61} - 1$). A length-n subfield-VOLE leaves P_1 with the global MAC key $\Lambda \in \mathbb{K}$ and $v \in \mathbb{K}^n$ and P_0 with $u \in \mathbb{F}_p^n$, $w \in \mathbb{K}^n$ satisfying

$$w \;=\; u \cdot \Lambda + v,$$

so each coordinate carries an information-theoretic Λ-MAC on u_i: (w_i, v_i) is the Λ-authenticated share $\langle\!\langle u_i \rangle\!\rangle$. *Silent* means a short base VOLE on $k \ll n$ seed coordinates is expanded locally by a public dual-LPN generator: at regular noise weight t the seed stretches to n pseudorandom authenticated coordinates whose secrecy rests on learning-parity-with-noise at (n, k, t). The base VOLE comes from the same $(\mathcal{F}_{\text{prep}}, \mathcal{F}_{\text{coin}})$ hybrid the stack assumes; C3 adds the expansion, its authentication and derandomisation into typed correlations, and the one-time accounting binding each correlation to the inventory. Authentication uses a standard sacrifice (two authenticated products generated, one consumed to check the other), so a surviving Beaver triple $(\langle\!\langle a \rangle\!\rangle, \langle\!\langle b \rangle\!\rangle, \langle\!\langle c \rangle\!\rangle)$, $c = ab$, is correct except with probability $O(1/|\mathbb{K}|)$. One expansion emits 131072 authenticated triples at a measured ratio n/k of 16–32×; each triple, mask pair, and FSS key is stamped for one-time use, so no correlation is spent twice.

Algorithm 2 Producer–consumer preprocessing scheduler (run by each party P_b, $b \in \{0, 1\}$).

State: ready queues $\{Q_\tau\}$; consumed nonces C; reject log R; watermarks $\ell < h$; window W; public demand $\mathbf{d}$; plan π.

PRODUCER (logical thread): repeat until session is RELEASED
1 pick a type τ with $d_\tau > 0$ and least fill $|Q_\tau|/h$
2 **if** $|Q_\tau| < \ell$ **then** *(refill)*
3 $B \leftarrow$ next $\min(W,\, h - |Q_\tau|)$ jobs of type τ from π
4 $\{k_i\} \leftarrow \mathsf{GenTrace}(B)$ in the $(\mathcal{F}_{\text{prep}}, \mathcal{F}_{\text{coin}})$-hybrid *(VOLE-expanded)*
5 $v \leftarrow \mathsf{GenCheck}(\{k_i\})$ *(aggregated verdict)*
6 **if** $v = \mathsf{accept}$: stamp each k_i with a fresh nonce η_i under sid; enqueue to Q_τ
7 **else**: append $\{k_i\}$ to R; **abort**
8 **else** *($|Q_\tau| \geq h$)*: wait *(backpressure)*

CONSUMER (logical thread): on each planned online gate g of type τ
9 **require** session state $=$ online *(certification barrier passed)*
10 $k \leftarrow$ pop front of Q_τ; **assert** $\eta_k \notin C$; $C \leftarrow C \cup \{\eta_k\}$ *(one-time)*
11 $\langle\!\langle y \rangle\!\rangle \leftarrow \mathsf{Eval}(k, \langle\!\langle x \rangle\!\rangle)$; absorb (η_k, g) into the transcript
12 **at** release barrier: **if** every $\eta \in C$ has an accept verdict **then** Release **else abort**

The roles change *when* certified material is produced, never *whether* it is certified before use, letting us reason about overlap and pipeline security separately.

Proposition 1 (Ready-queue invariants)**.** *For every session admitted by the state machine of §5 and scheduled by Algorithm 2:* (i) *no item enters* ready *before* GenCheck *accepts its batch, so no uncertified key is consumable;* (ii) *each nonce is consumed at most once, an item whose $\eta \in C$ being refused;* (iii) *every producer decision (refill, prefetch size, backpressure) is a function of public quantities ($|Q_\tau|$, $\mathbf{d}$, ℓ, h, W) only;* (iv) Release *is reachable only after the release barrier, which requires an accept verdict for every consumed nonce, so release never crosses the certification barrier;* (v) *nonces are namespaced under* sid*, so items of one session are neither visible nor consumable in another;* (vi) *overlapping the roles does not change the transcript: the GenTrace/GenCheck/Eval/Release messages and their order are fixed by π and the barriers, and the scheduler only chooses the wall-clock time at which public-count-gated batches launch.*

Each clause reduces to an assertion on a public counter or nonce set checked at the corresponding transition, so violating inputs cause a refusal or abort, not a release. All six are enforced in code and exercised by a test suite injecting each violation (an uncertified ready-queue push, a replayed nonce, a demand-dependent refill, a premature release, a cross-session nonce, a reordered launch), confirming a reject or abort every time. The prior prefill/decode stage scheduler is not part of this pipeline; if retained it is only an optional sub-scheduler inside the consumer role, never a substitute for the certified inventory.

Four latencies, and what the overlap actually hides. Measuring the overlap honestly separates four quantities: the *offline latency* to generate and certify the whole inventory in isolation; the *online critical-path latency* of the consumer's dependent chain; the *overlapped latency*, the part of offline latency running concurrently with online work and so never on the critical path (what the pipeline maximises); and the *total resource cost*, the summed compute across threads, which overlap moves in time but does not reduce. Figure 8 reports the overlapped-over-serial speedup against queue depth and load regime: under balanced load the overlap hides preprocessing to a median $1.46\times$; the producer-slower regime stays between 1.06 and $1.18\times$ (the consumer waits on generation) and the consumer-slower regime reaches about $1.41\times$ (generation fully absorbed). Serial and overlapped runs execute identical jobs and gates with no work relocated outside the timing window, so the ratio measures scheduling, not accounting.

A Half-Tree backend for the read-only keys. The full-domain FSS-for-read construction underlying the LUT gather originates with Floram (Doerner & Shelat, 2017); the verification core exposes it as a read-only key with an Eval/full-domain interface where a DPF realises $\beta \cdot \mathbb{1}\{x = \alpha\}$. We add an adaptation layer presenting that interface but building the point-function key with Half-Tree (Guo et al., 2023), which halves the per-level GGM cost: rather than two seed-correction words per level, one child seed is *derived* from a correlation-robust hash and only one correction word is sent. The certification path is untouched, since the key still evaluates through the same read-only API. Across point-function domains of size $n \leq 14$ the Half-Tree key is functionally equivalent to the reference DPF ($\beta \cdot \mathbb{1}\{x = \alpha\}$ at every x) at a measured key ratio 0.52, just over half the correction words. It costs one extra assumption: a circular correlation-robust hash keyed by a fixed offset in the GKWY20 style, no weaker than the fixed-key AES the stack already uses and confined to the DPF backend; comparison (DCF, $\beta \cdot \mathbb{1}\{x < \alpha\}$) keys stay on the reference path.

GPU kernels: CUDA PRGs and a limb-decomposed CUTLASS ring GEMM. The online arithmetic is regular and data-parallel, so we move it to the GPU without changing what is computed. NVRTC-compiled CUDA kernels run the PRG/AES tree expansion, the DPF/DCF full-domain evaluation, the gathered LUT contraction, and the ring add/multiply modulo 2^k (k=64; a Q21.16 payload spans 38 bits of its 64-bit residue, so no operand fits a native 8-bit type). The linear layers therefore run as an exact *limb-decomposed* ring GEMM: a residue splits into eight 8-bit limbs $a = \sum_i a_i 2^{8i}$, a product expands to $\sum_{i,j} a_i b_j \, 2^{8(i+j)}$, and only the 36 limb pairs with $8(i{+}j) < 64$ survive. Each pair is one CUTLASS int8 tensor-core GEMM (an unsigned limb u travels as the signed byte $u - 128$, with the zero-point's contribution, $128 \cdot$ row sums $+ 128 \cdot$ column sums $+ 128^2 K$, added back) whose partial products accumulate in int32, widen to int64, and add under the limb shifts. Modular reduction costs nothing: the discarded pairs plus the wrap of 64-bit addition *are* arithmetic modulo 2^{64}; sign recovery is deferred to the fixed-point decode of the two's-complement residue. A coarser radix-2^{16} variant (10 surviving pairs on the fp64 pipeline) issues $3.6\times$ fewer partial GEMMs and delivers the resident throughput below. Every kernel and both radices are validated bit-exactly against the CPU reference (overflow, modular reduction, sign handling, fixed-point rounding, and determinism all match), letting the GPU and CPU paths certify against the same ledger. Reported timings include H2D/D2H transfer and kernel launch, not just compute. With operands resident on the device the ring GEMM reaches 57 GMAC/s against 9.8 GMAC/s on the CPU, a $5.8\times$ speedup, one MAC counting one *complete* $\mathbb{Z}_{2^{64}}$ *ring MAC* (MNK per GEMM), never a limb-level int8 MAC (10–36 limb MACs are issued per ring MAC). A single isolated call gains only 0.7–$2.1\times$ because transfer dominates, so the full speedup appears only when a sequence of layers amortises residency, the regime the model runs in.

Together the four components answer Q3: $\mathcal{F}_{\text{prep}}$ is instantiated from silent subfield-VOLE, expanded 16–$32\times$ into one-time typed correlations; the producer–consumer inventory certifies that material ahead of demand and hands it across a certification barrier, hiding a median $1.46\times$ of the dealer-free preprocessing behind online work while preserving the six invariants of Proposition 1; the Half-Tree backend cuts read-only key material to a 0.52 ratio under one well-understood assumption; and the CUDA/CUTLASS kernels run the online arithmetic bit-exactly at up to $5.8\times$ on resident data, so the online path never waits on uncertified material nor pays the dealer-free generation cost on its critical path.

H EXTENDED SECURITY ANALYSIS

The core certifies one gate, but an inference is a whole session: it fixes an approximation plan, produces and certifies key material in the background, evaluates thousands of gates and linear layers against a ready queue, and opens one authenticated output. We show that the runtime (C1), the cost-aware plan (C2), and the preprocessing pipeline (C3) add no adversarial capability and leak nothing beyond the core's, each reducing to the inherited gate guarantee, a function of public data, or a transcript-preserving transformation.

Inherited guarantee. The core operates in the $\mathcal{F}_{\mathrm{prep}}$-hybrid model against one statically corrupted, rushing, malicious party with abort; we reuse three of its results (Xue et al., 2026). Its MAC keys, Δ on offline wires ($[\![\cdot]\!]$) and Λ on FSS outputs ($\langle\!\langle\cdot\rangle\!\rangle$), bind offline and online material into one authenticated domain; gate-lifting gives active security at no online cost to every gate expressible as public affine post-processing of a constant number of DPF/DCF evaluations; and GenTrace, GenCheck, and the aggregated-Λ-MAC opening Release certify a key with soundness error at most $\approx 2n/|E|$ in the audit algebra $\mathbb{K}$ (n being the domain bit-length and E the public challenge set), subject only to the previously specified abort convention.

The following core argument accompanies Theorem 3; the full proof follows in Appendix I.1.

Proof (core). W.l.o.g. let P_1 be corrupted (the other case is symmetric). The simulator $\mathcal{S}$ (i) emulates $\mathcal{F}_{\mathrm{prep}}$/$\mathcal{F}_{\mathrm{coin}}$, learning every correlation handed to $\mathcal{A}$; (ii) runs GenTrace/GenCheck honestly on dummy parameters: the transcript is distributed identically because correction-word openings are one-time-padded by PRG outputs; (iii) answers every online masked opening $\hat{x} = x + r_{\mathrm{in}}$ with a fresh uniform ring element, exactly the real distribution since each r_{in} is uniform and consumed once (queue invariants, Proposition 1); and (iv) extracts $\mathcal{A}$'s input from its $\mathcal{F}_{\mathrm{prep}}$ handles, queries $\mathcal{F}_{\mathrm{inf}}$, and patches the final opening to the ideal output. Any deviation by $\mathcal{A}$ either flips a certification verdict (caught by GenCheck except with probability $\varepsilon_{\mathrm{gen}}$ per instance) or shifts a released share (caught by Release except with $\varepsilon_{\mathrm{rep}}$); the state machine makes these the only two exits, since use-before-certify is refused and release is unreachable before the barrier. Summing over m instances and one aggregated release gives the bound. The full simulation, the hybrid argument for the masked openings, and the correctness/privacy cases are given in Appendix I. □

The approximation plan is public (C2). The cost-aware compiler emits a per-operator, per-region plan over lookup tables and low-degree splines that is a deterministic function of the public architecture and accuracy budget δ alone, depending on operator curvature and the active-verification price of a degree but on no activation value. Settled in the PLANNED state before the first key request, its approximation error is a matter of correctness, not confidentiality: it induces no secret-dependent transcript variation and so is no leakage channel.

Scheduling adds no secret-dependent behaviour (C3). The producer–consumer scheduler's six queue invariants (no uncertified key enters the ready queue, each is consumed once, refill counts depend only on public shapes, release never crosses the certification barrier, sessions are isolated, and overlapping the roles leaves the transcript unchanged) make the demand on $\mathcal{F}_{\mathrm{prep}}$, the dispatch order, and the release point functions of public data (Proposition 1), so concurrent prefetch-and-certify opens no secret-dependent channel. The Half-Tree backend replaces the point-function keys with an equivalent, more compact form (verified for $n \leq 14$), changing only the offline key representation and adding one assumption: a circular correlation-robust hash keyed by a fixed offset. The CUDA/CUTLASS kernels (PRG expansion, LUT gather, ring operations, int8 ring GEMM) are deterministic and bit-exact against the CPU reference, so they change no message.

Additional assumptions and trust boundaries. DUPLEX inherits the core's fixed-key AES assumption and adds three localised items. (i) The silent-VOLE producer instantiating $\mathcal{F}_{\mathrm{prep}}$ rests on a dual-LPN assumption over the working ring (Boyle et al., 2019a;b); proved in the $\mathcal{F}_{\mathrm{prep}}$-hybrid, online security invokes it only when $\mathcal{F}_{\mathrm{prep}}$ is realised, never on the online transcript. (ii) The Half-Tree hash is a circular correlation-robust primitive keyed by a fixed offset (GKWY20-style; Guo et al., 2023), no weaker than the fixed-key AES already assumed. (iii) The GPU is a trusted-computing boundary *within* one party (it holds only that party's shares and opens no channel to the peer), so

its security role reduces to kernel determinism, checked bit-exactly against the CPU. Side channels (timing, microarchitectural contention, device-level leakage) are out of scope.

These reductions establish that DUPLEX realises the declared inference functionality with abort under one static malicious corruption, subject precisely to the core's prescribed verification-triggered abort convention and assuming fixed-key AES plus the three additions, with full proofs and simulators provided in Appendix I.

I DEFERRED PROOFS

This appendix collects the arguments deferred from the body: the optimality of the segmentation dynamic program (Theorem 2) and the full simulation proof of the main theorem (Appendix I.1). The per-gate hybrids they rest on (the gate-lifting theorem, the soundness of GenTrace/GenCheck, and the aggregated Λ-MAC Release) are inherited verbatim from the verification core (Xue et al., 2026) and are not reproduced; what is new to DUPLEX is how those pieces compose and how the session binds them. For reference, Table 3 restates the inherited core interface and Table 4 the adopted security model (both referenced from §2), and Algorithm 3 restates the online execution loop of Section 5.

Optimal substructure of the segmentation DP. A plan partitions the operator's input range into consecutive regions over the grid atoms $x_0 < \cdots < x_N$ of Definition 1. Each region carries a mode $m \in \{\mathrm{TAB}\} \cup \{\mathrm{SPL}_d : 1 \le d \le D\}$, and the plan cost is the sum of per-region prices plus at most two *fixed bundle* charges, one for TAB and one for SPL, each levied once if any region uses that family. Let $s \in \{0,1\}^2$ record which bundles are already paid and let $C[i,s]$ be the minimum cost of covering the suffix $[x_i, b)$ in bundle-state s, exactly as in Figure 6. A region starting at x_i ends at some later atom x_j, and its incremental price depends only on (i, j, m) together with the single bit of

Algorithm 3 Online execution under session sid (party P_b, $b \in \{0,1\}$), from Section 5.

Require: state = ONLINE; certified inventory $\mathcal{I}$ bound to sid; Λ-authenticated input $\langle\langle x_0 \rangle\rangle$
for $\ell = 1, \ldots, L$ **do**
  if layer ℓ is linear **then**
    $\langle\langle x_\ell \rangle\rangle \leftarrow \textsc{RingGEMM}(\langle\langle x_{\ell-1} \rangle\rangle, W_\ell)$ ▷ exact limb-decomposed $\mathbb{Z}_{2^{64}}$ ring GEMM, authenticated Beaver openings
    $\langle\langle x_\ell \rangle\rangle \leftarrow \mathsf{Eval}(\text{trunc gate})$ ▷ truncation as certified offset gate
  else ▷ FSS nonlinear layer, certified gate $G^{[r_{\text{in}}, r_{\text{out}}]}$
    assert verdict for this key = accept and key unconsumed
    open $\hat{x} \leftarrow x_{\ell-1} + r_{\text{in}}$ ▷ uniform over the ring, hides $x_{\ell-1}$
    $\langle\langle x_\ell \rangle\rangle \leftarrow \mathsf{Eval}(F_{\theta,\Lambda}, \hat{x}) = G(\hat{x} - r_{\text{in}}) + r_{\text{out}}$
    mark key consumed; absorb queue entry into h_t
  end if
end for
$y \leftarrow \mathsf{Release}(\langle\langle x_L \rangle\rangle)$ ▷ aggregated Λ-MAC check over ONLINE outputs
if check passes **then** state $\leftarrow$ RELEASED; **return** y
else abort (no release)
end if

Table 3: Inherited verification-core interface (Xue et al., 2026) (n: domain bits; m: batch size; $E \subseteq \mathbb{K}$: challenge set).

Primitive	Role	Cost order
$F_{\theta,\Lambda}$	Two-plane key; each eval is a Λ-authenticated share	$O(\lambda n)$ size
GenTrace	Dealer-free distributed key generation	$O(\lambda n)$/key
GenCheck	Batch certification via six-residual ledger	$O(m)$, $\approx 2n/\lvert E\rvert$
Eval	Local full-domain / single-point evaluation	$O(2^n)$ / $O(n)$
Release	Open through one aggregated Λ-MAC	$O(1)$
Aggregate	Cross-domain join of certified shares	$O(1)$

s that records whether m's bundle has been paid; it is independent of how $[x_j, b)$ is covered. Hence the suffix of an optimal plan is itself optimal for (j, s'), where s' is s with m's bundle bit set: if a cheaper cover of $[x_j, b)$ existed, substituting it would strictly lower the total, contradicting optimality. This exchange argument is Bellman optimality, so the backward recursion

$$C[i, s] = \min_{j>i,\ m:\ \mathrm{err}(i,j,m)\le\delta} \mathrm{price}(i, j, m \mid s) + C[j, s']$$

with base case $C[N, \cdot] = 0$ returns a global optimum at $C[0, \varnothing]$.

Sufficiency of the two-bit state. The only coupling between non-adjacent regions is the once-only fixed cost of each family; two bits record exactly that coupling, and no finer state is needed because prices are otherwise additive across regions. The state space is therefore $4(N+1)$; each state ranges over $O(N)$ endpoints j and $D+1$ modes, and each transition is $O(1)$ given the precomputed feasibility oracle, giving the $O(N^2(D+1))$ time and $O(N)$ memory bound of Theorem 2. The plan then goes through compile–measure–tighten (Appendix J): optimality here is optimality of the *priced* plan, and feasibility is re-established by execution rather than by the cost model.

I.1 Full Proof of Theorem 3

Theorem 5 (Security of Π_{Duplex}; restated from Theorem 3)**.** *In the $(\mathcal{F}_{\mathrm{prep}}, \mathcal{F}_{\mathrm{coin}})$-hybrid model, Π_{Duplex} (Algorithm 1) securely realises $\mathcal{F}_{\mathrm{inf}}$ with abort against one statically corrupted, rushing, malicious party, under the verification core's prescribed abort convention, with distinguishing advantage at most $m\,\varepsilon_{\mathrm{gen}} + \varepsilon_{\mathrm{rep}} + \mathsf{negl}(\lambda)$.*

Proof. Preprocessing is handled by the $\mathcal{F}_{\mathrm{prep}}$-hybrid: every correlation, mask and triple the corrupted party receives is emulated by the simulator, so the silent-VOLE realisation of $\mathcal{F}_{\mathrm{prep}}$ enters only through its own (dual-LPN) reduction and never through the online transcript. W.l.o.g. let P_1 be the corrupted party $\mathcal{A}$; the case of a corrupted P_0 is symmetric. The simulator $\mathcal{S}$ proceeds as follows.

(1) $\mathcal{S}$ emulates $\mathcal{F}_{\mathrm{prep}}$ and $\mathcal{F}_{\mathrm{coin}}$, recording every correlation, MAC key share and public coin handed to $\mathcal{A}$.

(2) $\mathcal{S}$ fixes the public plan $\pi = \mathsf{Compile}(M, \delta)$ honestly and absorbs $(\mathsf{sid}, \pi, \mathcal{I})$ into the transcript hash; all quantities are public, so this step is identical in both worlds.

(3) For every job in $\mathcal{I}$, $\mathcal{S}$ runs GenTrace with $\mathcal{A}$ on dummy parameters (θ'_j). Each correction-word opening is one-time-padded by a fresh PRG output, so $\mathcal{A}$'s view of generation is computationally indistinguishable from the real one (this is the core's generation hybrid; Xue et al., 2026).

(4) $\mathcal{S}$ runs GenCheck under one aggregated challenge from $\mathcal{F}_{\mathrm{coin}}$. If $\mathcal{A}$ deviated in generation, $\mathcal{S}$ learns the deviation from its $\mathcal{F}_{\mathrm{prep}}$ emulation; it submits the corresponding verification condition to $\mathcal{F}_{\mathrm{inf}}$ and aborts exactly when the predicate fires, matching the real abort distribution up to $\varepsilon_{\mathrm{gen}}$ per instance.

(5) In the online phase, $\mathcal{S}$ answers every masked opening $\hat{x} = x + r_{\mathrm{in}}$ with a fresh uniform ring element: each r_{in} is uniform, secret and consumed exactly once (the queue invariants of Proposition 1 forbid reuse and use-before-certify), so real and simulated openings are identically distributed. Linear layers exchange only Beaver-masked differences, likewise uniform.

Table 4: Security model adopted by Duplex, inherited from Xue et al. (2026).

Goal	Confidentiality of prompt, weights, and activations; correct output or abort.
Adversary	One statically corrupted, malicious, rushing party.
Guarantee	Security with abort; deviation caught before Release.
Hybrid	$\mathcal{F}_{\mathrm{prep}}$ (correlated randomness) and $\mathcal{F}_{\mathrm{coin}}$ (public coins).
Confidentiality	The LLM's weights and users' prompt information remain confidential throughout the protocol.
Out of scope	Collusion beyond one corruption, denial of service, traffic/timing channels beyond the public shapes, and inference drawn from correctly released outputs.

(6) $\mathcal{S}$ extracts $\mathcal{A}$'s input share from its recorded $\mathcal{F}_{\text{prep}}$ handles, queries $\mathcal{F}_{\text{inf}}$ for the output y, and patches the final aggregated opening so that the released value equals y, using its knowledge of $\mathcal{A}$'s Λ-MAC shares.

(7) If $\mathcal{A}$ shifts any released share, $\mathcal{S}$ lets the emulated Release check fail and aborts; by the core's report-binding lemma this differs from the real world with probability at most ε_{rep}.

Correctness. On an honest run every gate output satisfies the two-plane invariant $h = \Lambda \cdot x$; certified truncation and the exact ring GEMM preserve it, so the released output equals the plan's fixed-point value of $M(x; W)$ within the budget δ (a correctness, not a security, property; cf. Appendix J).

Privacy. Steps (2), (3) and (5) exhaust $\mathcal{A}$'s view: public plan material, one-time-padded generation openings, and uniform masked openings. A standard two-step hybrid shows the simulated view is indistinguishable: Hyb_0 the real view; Hyb_1 replacing generation openings by PRG-padded dummies (loss $\mathsf{negl}(\lambda)$ by the core's generation hybrid); Hyb_2 replacing masked openings by uniform elements (loss 0, perfect).

Active security. The session state machine leaves exactly two exits: a deviation before the certification barrier must survive GenCheck (probability $\leq \varepsilon_{\text{gen}}$ per instance, union-bounded over m instances under one aggregated challenge), and a deviation after it must survive the aggregated Λ-MAC of Release (probability $\leq \varepsilon_{\text{rep}}$). Use-before-certify is refused and Release is unreachable before the barrier, so no third exit exists. Collecting terms,

$$\mathsf{Adv}_{\mathcal{Z}} \;\leq\; m\,\varepsilon_{\text{gen}} \;+\; \varepsilon_{\text{rep}} \;+\; \mathsf{negl}(\lambda),$$

with $\varepsilon_{\text{gen}} \approx 2n/|E|$ and $\varepsilon_{\text{rep}} \leq 2/|E|$ over the audit algebra $\mathbb{K}$ (Xue et al., 2026). Whether the protocol aborts is governed solely by the aggregated condition in step (4). □

The per-gate hybrids this proof invokes (generation, certification soundness and report binding) are proved in the verification core (Xue et al., 2026) and are used unchanged; what is new is the session composition above, which we summarise component by component. The *inherited verification core* carries the gate-level obligations (active security of every gate and sound certification of every key, at soundness $\approx 2n/|E|$ over $\mathbb{K}$; Xue et al., 2026) and governs the protocol's prescribed abort behavior; its enforcing invariants are the two-plane authenticated key $F_{\theta,\Lambda}$, the aggregated Λ-MAC opened by Release, and the refusal of use-before-certify. The *cost-aware compiler* (C2) owes correctness and secret-independence of the plan and leaks nothing, because the plan is a deterministic function of the public architecture and δ, fixed in the PLANNED state before any key request is admitted. The *producer–consumer queue* (C3) must introduce no secret-dependent refill, reuse, early release, or transcript variation; this is exactly what the queue invariants of Proposition 1 enforce (no uncertified key in the ready queue, one-time consumption, refill from public counts only, release gated by the certification barrier, session isolation), and demand and order are public. The *Half-Tree backend* (C3) owes only equivalence to the point-function keys it replaces, resting on Half-Tree correctness (Guo et al., 2023) under a fixed-offset CCR hash and tested exhaustively for $n \leq 14$. Finally, the *CUDA/CUTLASS kernels* (C3) owe functional equivalence to the CPU reference: every kernel (PRG, LUT gather, ring ops, int8 GEMM) is deterministic and checked bit-exactly against the CPU on every run, and the device sits inside one party's trust boundary, so the kernels are transcript-preserving and add no leakage.

J Operator Compilation Details

The compiler reduces each nonlinearity to a bounded interval before fitting, so that a short table or low-degree spline suffices and the offset-gate tails carry the rest exactly. This appendix records the four reductions and the realised errors. All arithmetic is fixed-point Q21.16 (resolution 2^{-16}), and error is reported in units in the last place (ULP) at that resolution.

GELU. $\mathrm{GELU}(x) = x\,\Phi(x)$ saturates: for $x \gtrsim R$ it equals x to within one ULP, and for $x \lesssim -R$ it equals 0. The reduction clamps the fit to the finite band $[-R, R]$; the upper tail is realised by an offset gate returning x and the lower tail by the zero share, so only the band consumes a spline bundle.

Softmax. Over a row of S logits, the reduction subtracts the row maximum $m = \max_i x_i$ (an oblivious maximum over the authenticated scores, computed with the same DCF comparison used by the router), so every argument to the exponential is ≤ 0. The transcendental is then e^y on the half-bounded interval $y \in [-B, 0]$, with B chosen so that logits below $m - B$ underflow to 0 at Q21.16; normalisation is a single authenticated reciprocal of the sum. The subtraction of m is a public-offset map on authenticated shares and adds no gate.

SiLU. $\mathrm{SiLU}(x) = x\,\sigma(x)$ with σ the logistic sigmoid. Using $\sigma(x) = 1 - \sigma(-x)$, the reduction folds the domain to $x \geq 0$ on the authenticated sign bit, fits σ on $[0, R]$, and reconstructs by the fold; the tails are those of GELU ($\to x$ above R, $\to 0$ below $-R$). This is the activation inside the SwiGLU block of Llama.

LayerNorm and RMSNorm. The sole nonlinearity is an inverse square root: of the variance for LayerNorm, of the mean square for RMSNorm. The reduction normalises the positive input v by its binade, writing $v = 2^{2e}u$ with an even exponent $2e$ extracted by a DCF leading-bit scan and mantissa $u \in [1, 4)$; $u^{-1/2}$ is fitted once on $[1, 4)$ and the result is rescaled by 2^{-e}. LayerNorm additionally subtracts the mean beforehand, a public linear map on the shares.

Compile–measure–tighten. For each operator DUPLEX compiles a candidate plan, measures the realised error against a `float64` reference over the whole represented domain, and re-solves with a tightened fitting budget whenever any sampled point exceeds the target. All four operators converge to ≤ 1 ULP under this loop, so feasibility rests on measurement rather than on an analytic bound. Under active security the priced plan shifts the chosen spline degree *down* relative to the semi-honest plan (for one operator from 3 to 2), because each additional degree carries verification cost; the divergence between the two plans is modest in magnitude. A spline-only plan that ignores the table option costs $3.2\times$ more than the cost-aware plan under the active security model.

Relation to prior table and spline gates. Three lines of work fix separately what this compiler decides jointly. Pika (Wagh, 2022) reads a public function at a secret index through one DPF (a public-LUT, private-index gate), committing to the table mode under a dealer-assisted $(2{+}1)$-party model (semi-honest, or malicious with an honest majority), without choosing table versus spline or pricing dealer-free certification. Grotto (Storrier et al., 2023) builds low-degree splines from one DPF via prefix parities in the $(2{+}1)$-party setting, committing to the spline mode with a helper and semi-honest costs. FABLE (Su et al., 2025) does batched two-party lookups whose table is a *server secret*, a confidential-table setting orthogonal to a public-operator catalogue, not a general LUT baseline. DUPLEX's compiler is mode-agnostic: it treats table and spline as two affine, Λ-authenticated gate families whose outputs add, selecting per region under a cost model that prices the dealer-free active overhead (second authentication plane, full-domain certification pass, per-degree authenticated multiplication) that all three prior designs leave unpriced.

K ADDITIONAL EVALUATION

DUPLEX is implemented in C++ and Python over the inherited verification core, with CUDA kernels via NVRTC and an exact ring GEMM via CUTLASS; the compiler, runtime, PCG/VOLE producer, producer–consumer scheduler, Half-Tree backend and model executors are new, and the FSS core is used unchanged.

K.1 RQ1: IS THE DEALER-FREE ACTIVE PIPELINE CORRECT AND DOES IT DETECT DEVIATIONS?

We run the runtime under (i) honest execution and (ii) an execution in which a corrupted party tampers with a released share. Table 5 states the executed model, scale, topology, network setting, measured time, traffic, repetitions and checks. The honest run crosses the certification barrier and releases; the tampered run fails the aggregated Λ-MAC check and aborts with no output; use-before-certify is refused. A gate-level injected-deviation campaign rejects every injected deviation while accepting every honest generation: deviations are caught at the certification and release barriers rather than silently corrupting the output.

Table 5: The two-role C1 experiments behind RQ1 (median of five runs; platform of Appendix L).

Pipeline runs (honest / tampered / use-before-certify)	
Model	Three-layer MLP, 20×16 ring GEMM $\rightarrow$ certified ReLU $\rightarrow$ 16×4 GEMM, batch 4 (384 weights, 2 FSS instances); 10-bit gate domain, 12-bit scale, $\mathbb{Z}_{2^{64}}$ shares; the six LLMs of Section 8.3 reuse this software stack at 12-bit gate domains and a 16-bit scale (Q21.16).
Topology	Roles P_0, P_1 on two Appendix-L hosts; instrumented authenticated channel, exact byte/round accounting; no delay injected (latency projections reuse Table 11: LAN 0.05 ms/23.5 Gb/s, MAN 6 ms/1 Gb/s, WAN 50 ms/200 Mb/s).
Cost	Offline (keygen + certify) 0.44 s; online 46.3 ms (0.7 linear, 45.6 FSS); tampered run identical to the release check; use-before-certify refused in <0.01 ms. Traffic 15 072 B / 92 rounds per session (9 984 B/48 gen., 5 088 B/44 cert.).
Repet.	5 per scenario, fixed seeds; 36 injections (6 classes $\times$ DPF/DCF $\times$ 3); 5 honest generations. Honest: RELEASED, error 0.013; tampered (+7 on a released share): ABORTED, no output.
Checks	18-check suite, all pass: 9 core (algebra, gates, operators, soundness, GEMM, DP, composition) + 9 system (runtime, VOLE, scheduler, Half-Tree, GPU, model); itemised in Table 7.

K.2 RQ4: What do Half-Tree, CUDA and CUTLASS contribute?

These are validated as components, not folded into the end-to-end latency. Half-Tree brings seed-correction key material to 0.52 of the full-tree backend while preserving DPF point-function equivalence; the PCG/VOLE producer expands a short base VOLE by up to $32\times$ into one-time authenticated correlations. On the GPU the exact secret-shared ring GEMM runs at 57 vs. 9.8 GMAC/s ($5.8\times$) with device-resident operands (full ring MACs, not limb-level int8 MACs; Section 6), and every GPU primitive matches the CPU reference bit for bit, including overflow, modular reduction, sign extension and rounding. Figure 9 breaks the gain down: PRG expansion $5.3\times$, LUT contraction $4.6\times$, resident ring GEMM $5.8\times$, while a lone elementwise ring multiplication is transfer-bound at $0.6\times$, the honest boundary of what the GPU buys; a single isolated GEMM call is likewise only 0.7–$2.1\times$.

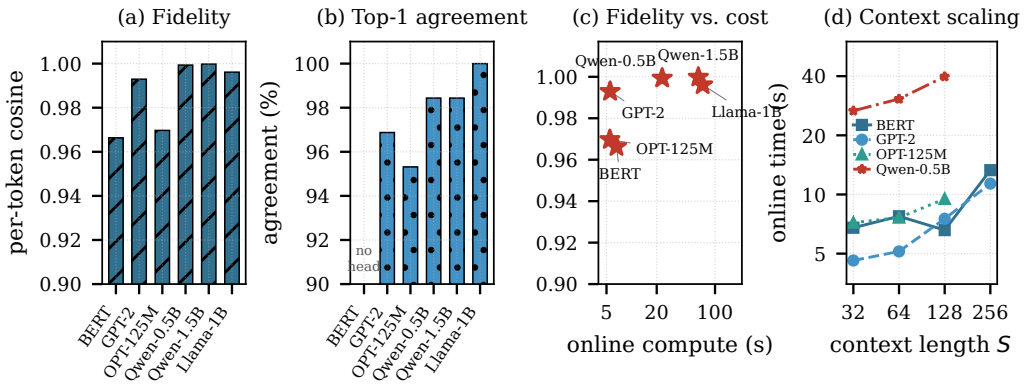


Figure 7: Fidelity (a) and next-token top-1 agreement (b) across the six checkpoints, fidelity versus executed online compute (c), and the scaling of the online time with the context length S for four checkpoints (d; measured sweep of Table 10). BERT-Base carries no task head, so only its cosine is reported.

L Artifact Appendix

L.1 Description & Requirements

Duplex is a dealer-free, actively secure two-party inference runtime built on an inherited FSS verification core. It couples a session state machine that makes uncertified consumption structurally impossible (C1) with a cost-aware LUT/spline compiler (C2) and a PCG/VOLE producer–consumer scheduler with Half-Tree and CUDA/CUTLASS backends (C3), and executes six real open checkpoints end to end (C4). The artifact covers the measurement campaigns, the raw logs, and every figure, table and headline number this paper derives from them.

L.1.1 How to access

Artifacts will be released as anonymized supplementary material with this submission.

L.1.2 Hardware dependencies

Any x86-64 CPU with AES-NI support can reproduce the C1 and C2 results, the C3 correctness results, and all deterministic metrics, including communication, round counts, triples, key sizes, and ULP errors. The C3 GPU throughput and bit-exactness experiments additionally require an NVIDIA CUDA-capable GPU. All experiments were conducted on two machines equipped with dual-socket Intel Xeon Platinum 83xx CPUs, 48 vCPUs, 512 GB of DDR4 memory, and two NVIDIA A100 SXM4 GPUs (80 GB each). Approximately 15 GB of storage is required for the model checkpoints.

L.1.3 Software dependencies

Experiments were conducted on Ubuntu 22.04 LTS with Python $\geq$3.9, NumPy, SciPy, matplotlib, and `cryptography`. The C++ components, including the EzPC secure-computation library, were built with g++-11 and CMake $\geq$3.22. GPU and model experiments additionally use PyTorch 2.0.1, CuPy, and `transformers` with CUDA 11.8 and a compatible NVIDIA driver. No network access is required after the model checkpoints have been downloaded.

L.1.4 Benchmarks

(1) Models: six public HuggingFace checkpoints (BERT-Base, GPT-2, OPT-125M, Qwen2.5-0.5B, Qwen2.5-1.5B and Llama-3.2-1B), each pinned to a released snapshot. (2) Data: a fixed public prompt set; no proprietary data. Each host is calibrated once before measurement and the resulting unit costs are carried with the raw logs.

L.2 Major Claims

- (C1): DUPLEX's dealer-free active pipeline executes end to end, releases under authentication on honest runs, and aborts with no output under injected deviation. This is proven by experiment (E1), with results reported in Appendix K.1 and Table 6.
- (C2): the cost-aware DP meets the accuracy budget and avoids up to $3.2\times$ mode penalty. This is proven by experiment (E2), with results reported in Figure 2 and Figure 3.
- (C3): producer–consumer overlap hides preprocessing, Half-Tree halves seed-correction key material, and the GPU kernels are bit-exact at $5.8\times$ resident GEMM speedup. This is proven by experiment (E3), with results reported in Figure 8 and Figure 9.
- (C4): six real checkpoints execute through the certified-operator stack with per-token cosine 0.9663–0.9998. This is proven by experiment (E4), with results reported in Table 1, Figure 7 and Table 9.

L.3 Evaluation

Table 6 maps each claim to its expected output and tolerance; the individual experiments follow.

L.3.1 Experiment (E1)

[pipeline correctness and abort] [2 human-minutes + 3 compute-minutes]: this experiment is designed to evaluate C1: an honest execution certifies and releases, an injected deviation aborts with no output, and use-before-certify is refused.

[Preparation] No additional GPU is required.

[Results] All 18 correctness checks pass: the honest session traverses $\mathsf{OPEN}\rightarrow\cdots\rightarrow\mathsf{RELEASED}$, the tampered session ends in ABORT with no released output, and a use-before-certify call is refused. The experiment is deterministic, so no tolerance applies. Table 7 itemises the suite (referenced from Table 5).

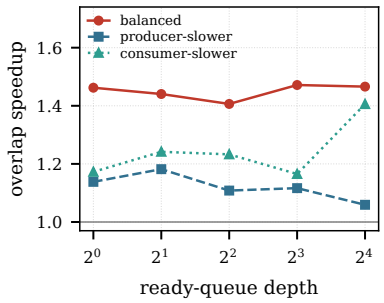


Figure 8: Producer–consumer overlap speedup across queue depths and load regimes (Section 8.2).

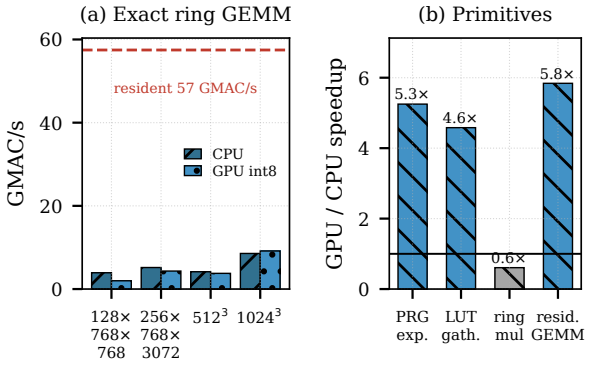


Figure 9: (a) Exact ring GEMM throughput per shape, CPU vs. GPU; (b) GPU/CPU speedup of the individual primitives. Timings include transfer and launch (Appendix K.2).

Table 6: Claims, expected outputs and tolerances for the artifact.

Claim	Expected output	Tolerance
C1	Honest run traverses OPEN$\rightarrow\cdots\rightarrow$RELEASED and certifies then releases; an injected deviation drives the session to ABORT with no release; a use-before-certify call is refused.	Exact: all assertions pass; no output is released on abort.
C2	GELU/Softmax/SiLU/LayerNorm each compile to ≤ 1 ULP; the spline-only plan costs $3.2\times$ the cost-aware plan; the active spline degree is $\leq$ the semi-honest degree.	ULP exact; cost ratio within $\pm 5\%$; degree relation exact.
C3	Silent VOLE relation $w = u\Lambda + v$ holds and expands 16–32×; all six scheduler invariants hold; Half-Tree matches the point function for $n \leq 14$ at key ratio ≈ 0.52; every GPU kernel is bit-exact against CPU; resident ring GEMM reaches $\approx$57 vs. 9.8 GMAC/s.	Correctness bit-exact; expansion in $[16, 32]$; key ratio ± 0.02; throughput hardware-dependent, $\pm 10\%$.
C4	Released outputs vs. the real HF checkpoints: BERT-Base cosine 0.9663; GPT-2 cosine 0.9929, next-token top-1 96.9%; Llama-3.2-1B cosine 0.9961, top-1 100.0%.	Cosine ± 0.005; top-1 exact on the fixed eval set.

Table 7: The 18-check correctness suite behind E1 and Table 5 (9 core + 9 system checks; all pass).

1–4	Wrap identities (exhaustive, $n \leq 6$); spline gates of degree 0–3 correct in value and tag planes; compiled recip/exp2/sigmoid/rsqrt within 1.05 ULP; certified DPF/DCF over $\mathbb{F}_{p^2}$, GF(2^{128}) and GR($2^{32}, 8$).
5–6	$36/36$ injected deviations (6 classes × DPF/DCF × 3 trials) rejected; $5/5$ honest generations accepted.
7–9	Limb-decomposed ring GEMM exact at three shapes; segmentation matches the brute-force optimum; a full MLP inference certifies and releases.
10–12	Honest session reaches RELEASED; tampered release share aborts with no output; use-before-certify is refused.
13–15	VOLE relation $w = u\Lambda + v$, triple relation and one-time consumption; all six queue invariants hold; overlap hides preprocessing (ratio >1.05).
16–18	Half-Tree $\equiv$ reference DPF ($n \in \{6, 8, 10, 12\}$, key ratio <0.6); CPU/CUDA/CUTLASS bit-exact including the int8 ring GEMM; a one-layer BERT-Base forward executes through the certified stack.

L.3.2 EXPERIMENT (E2)

[cost-aware compiler] [2 human-minutes + 4 compute-minutes]: this experiment is designed to evaluate C2 by compiling the four mandatory operators under every baseline mode and the cost-aware DP.

[Preparation] None.

[Results] The compilation campaign produces the data behind Figure 2 and Figure 3. Realised error is ≤ 1 ULP (exact); cost ratios may vary by about $\pm 5\%$ with host calibration, but the ordering of the modes and the chosen degrees remain consistent.

L.3.3 EXPERIMENT (E3)

[preprocessing pipeline and GPU] [2 human-minutes + 6 compute-minutes, GPU]: this experiment is designed to evaluate C3: silent-VOLE expansion, the scheduler invariants, Half-Tree equivalence, and the bit-exact GPU backends.

[Preparation] A CUDA build of PyTorch and cupy (the correctness half runs CPU-only).

[Results] The VOLE relation $w = u\Lambda + v$ holds and expands 16–32$\times$ (exact); all six scheduler invariants hold (exact); the Half-Tree key ratio is 0.52 ± 0.02; every GPU kernel is bit-exact against the CPU reference (exact). Resident GEMM speedup is 5.8$\times$ with a run-to-run range of about $\pm 5\%$; the overlap median varies by about $\pm 15\%$ across reruns.

L.3.4 EXPERIMENT (E4)

[end-to-end model campaign] [5 human-minutes + ≈30 compute-minutes]: this experiment is designed to evaluate C4 by running all six checkpoints through the certified-operator secure stack.

[Preparation] The one-time checkpoint download (≈15 GB of disk).

[Results] Each checkpoint yields one record giving the model, its utility (per-token cosine and top-1 agreement), the total communication in bytes and the linear and FSS online round counts: for GPT-2, a per-token cosine of 0.9929 at 0.969 top-1 agreement. Communication and round counts are exact, per-token cosine may vary by about ± 0.005, and compute latency is hardware-dependent ($\pm 10\%$).

L.3.5 REGENERATING THE PAPER'S NUMBERS

Every figure, generated table and headline macro of this paper is derived from the raw measurement logs, so a reproduced run propagates to the manuscript unchanged; the dealer-based BGI DPF online baseline of Table 12 is re-measured on the local host ($\pm 10\%$, best across runs).

L.4 EXTENDED RESULTS

L.4.1 MODEL CONFIGURATIONS

The end-to-end runs use released HuggingFace checkpoints with the configurations of Table 8, covering encoder and decoder architectures, multi-head and grouped-query attention, three activation families (GELU, ReLU, SiLU) and both normalisations; the table also lists the parameter count of each pinned snapshot and the per-forward operator inventory (ring MACs, Softmax and activation elements, norm rows) that the certified stack serves at the executed context length. All runs share the Q21.16 fixed-point format and the default audit algebra $\mathbb{F}_{p^2}$ with $p = 2^{61} - 1$.

L.4.2 EXTENDED PER-MODEL MEASUREMENTS

Table 9 gives the full per-model measurements behind Table 1: the offline/online split, the round and byte accounting of the linear and FSS parts, the certified-operator key material, the authenticated-triple count, the global and per-token fidelity, the mean relative error, and the modelled LAN/MAN/WAN totals (the structurally scaled communication at a full context of $S{=}512$ is the @512 column of Table 1 for the six headline checkpoints; the same fit gives 11.4 GB for TinyLlama-1.1B). Table 10 adds the measured sensitivity to the context length, together with the run-to-run spread of the timings.

Table 8: Executed checkpoints: architecture (public), parameter count read from the pinned HuggingFace snapshot, and the per-forward operator inventory the secure stack must serve at the executed context length S (all decoders use rotary positions, BERT/GPT-2/OPT learned ones; linear work in ring MACs; Softmax and activation elements are the scalar invocations of the certified gates; norm rows are the LayerNorm/RMSNorm rsqrt invocations; $H{=}q/kv$: grouped-query attention).

Model	Arch.	Params (M)	d	L	H	d_{ff}	Vocab	Norm / act.-FFN	S	MACs (G)	Softmax (M)	Act. (M)	Norm rows (K)
BERT-Base	Enc.	109.5	768	12	12	3 072	30 522	LayerNorm / GELU-MLP	128	14.2	2.36	4.72	3.2
GPT-2	Dec.	124.4	768	12	12	3 072	50 257	LayerNorm / GELU-MLP	64	8.0	0.59	2.36	1.6
OPT-125M	Dec.	125.2	768	12	12	3 072	50 272	LayerNorm / ReLU-MLP	64	8.0	0.59	2.36	1.6
Qwen2.5-0.5B	Dec. (GQA)	494.0	896	24	14/2	4 864	151 936	RMSNorm / SiLU-SwiGLU	64	31.8	1.38	7.47	3.1
Qwen2.5-1.5B	Dec. (GQA)	1543.7	1536	28	12/2	8 960	151 936	RMSNorm / SiLU-SwiGLU	64	99.1	1.38	16.06	3.6
Llama-3.2-1B	Dec. (GQA)	1235.8	2048	16	32/8	8 192	128 256	RMSNorm / SiLU-SwiGLU	64	79.4	2.10	8.39	2.1
TinyLlama-1.1B	Dec. (GQA)	1100.0	2048	22	32/4	5 632	32 000	RMSNorm / SiLU-SwiGLU	64	66.6	2.88	7.93	2.9

TinyLlama-1.1B appears here with its measured values: this checkpoint is intrinsically perturbation-sensitive (its own float forward drops to 0.74 cosine under 10^{-4} input noise), so its hidden-state fidelity degrades under any bounded-precision arithmetic; the row is a property of the checkpoint, not of the stack.

Table 9: Extended per-model measurements behind Table 1 (all seven executed checkpoints, evaluation host of Appendix L). Time: offline = dealer-free generation and certification of the operator keys, online = the secure forward. Rounds and bytes are the accounted transcript (lin. = Beaver openings of the ring GEMMs; FSS = key-generation plus certification traffic of the certified gates, whose online cost is zero bytes). Key = certified-operator key material; inst. = certified FSS instances. Fidelity is against the plaintext float model. LAN/MAN/WAN totals add the modelled transmission of the profiles of Table 11. TinyLlama-1.1B is intrinsically perturbation-sensitive (Section 8.3).

		Time (s)		Rounds		Comm.		Triples	Key		Fidelity				Total (s)		
Model	S	offl.	onl.	lin.	FSS	lin. (MB)	FSS (KB)	($\times 10^6$)	(KB)	inst.	global	token	rel. err	top-1	LAN	MAN	WAN
BERT-Base	128	18.7	6.6	360	325	812	119.0	312.6	8.9	16	0.9653	0.9663	0.31	n/a	6.9	17.2	73.3
GPT-2	64	18.4	5.5	336	325	731	119.0	80.8	8.9	16	0.9704	0.9929	0.17	96.9	5.8	15.3	67.8
OPT-125M	64	18.1	5.5	360	325	741	119.0	80.8	8.9	16	0.9705	0.9697	0.24	95.3	5.8	15.5	69.4
Qwen2.5-0.5B	64	19.9	23.2	840	745	3 032	119.0	195.6	8.9	16	0.9993	0.9994	0.06	98.4	24.4	57.0	223.8
Qwen2.5-1.5B	64	21.0	63.1	868	757	10 819	119.0	393.6	8.9	16	0.9998	0.9998	0.04	98.4	66.9	159.4	577.1
Llama-3.2-1B	64	21.6	71.1	1136	1073	8 020	119.0	292.6	8.9	16	0.9961	0.9961	0.09	100.0	73.9	148.5	502.3
TinyLlama-1.1B	64	19.8	63.6	1562	1475	8 045	119.0	394.3	8.9	16	0.6897	0.6935	0.83	6.2	66.5	146.2	537.2

L.4.3 Reading the communication figures

The online communication measured end to end is large (0.7–11 GB across the models) because it counts every Beaver opening per linear layer together with the authentication (tag) plane at full precision. This is the real cost of the stronger threat model rather than a tuning artifact, and the reproduction reports it unrounded so that it can be checked against the per-layer accounting. Table 11 decomposes the resulting end-to-end latency across LAN, MAN and WAN profiles: on a LAN the measured compute dominates, while on a WAN the transcript sets the total. Its lower rows re-express the representative baselines of Table 12 on the same three profiles at matched sequence lengths (BERT-Base at $S{=}128$, GPT-2 at $S{=}64$), each anchored to the system's own published LAN and WAN measurements; the SPDZ-style row is composed from the per-call costs measured on this host (7.08M nonlinear calls, 24-way parallel compute, 130 rounds per serial nonlinear stage). Table 12 compares one certified activation across the two-party baseline tiers of §8 and closes with three DUPLEX checkpoints from Table 11 at inference granularity.

L.4.4 Where the advantage is, and where it is not

Three advantages are marked in the two tables. First, the certified gate is the only row that is at once **dealer-free** and **actively secure** with **zero online bytes and zero rounds** per call: SIGMA's GELU opens 58 B over 7 rounds, Pika-style lookups pay their full-domain expansion online (17.0 ms measured on this host at the same 14-bit width, i.e. $118\times$ the certified gate), the client-malicious GC systems and BumbleBee pay ≈4–10 rounds, and the generic SPDZ-style route pays 8.3 KB over 130

Table 10: Sensitivity of the end-to-end execution to the context length S (evaluation host of Appendix L; same stack, 12-bit gate domains and accounting as Table 1; WAN total adds the modelled transmission of the accounted transcript at 50 ms/200 Mbps). Communication grows sub-linearly in S because the Beaver openings of the weight shares dominate, while the attention term grows quadratically and sets the @512 column of Table 1; the FSS round count is fixed by the layer structure. Fidelity is stable for GPT-2 up to S=256 and for the other decoders up to S=128, but BERT-Base falls to 0.14 at S=256: the context lengths of Table 1 are where the Q21.16 stack is validated, and long-context fidelity remains open (Appendix B). Repeatability: 3 GPT-2 runs at S=64 with distinct seeds give online 5.07–5.16 s (median 5.13 s, spread 1.9%) and offline 18.2–18.4 s, with identical transcript, rounds, triples and fidelity.

		Gate	**Time (s)**		**Comm.**		**Triples**	**Key**	**WAN**	**Fidelity**	
Model	S	bits	offline	online	(MB)	**Rounds**	($\times 10^6$)	(KB)	total (s)	Cos.	Top-1 (%)
(a) Context length S (12-bit gate domains)											
BERT-Base	32	12	17.7	6.8	709	685	21.5	8.9	69.4	0.9996	n/a
	64	12	18.6	7.7	741	685	80.8	8.9	71.6	0.9994	n/a
	128	12	18.7	6.6	812	685	312.6	8.9	73.3	0.9663	n/a
	256	12	18.0	13.3	982	685	1229.2	8.9	86.8	0.1352	n/a
GPT-2	32	12	18.6	4.6	704	661	21.5	8.9	65.8	0.9994	93.8
	64	12	18.4	5.1	732	661	80.8	8.9	67.4	0.9929	96.9
	128	12	19.0	7.5	793	661	312.6	8.9	72.3	0.9945	98.4
	256	12	18.0	11.4	944	661	1229.2	8.9	82.2	0.9925	99.2
OPT-125M	32	12	18.9	7.2	709	685	21.5	8.9	69.8	0.9597	90.6
	64	12	18.4	7.6	741	685	80.8	8.9	71.5	0.9697	95.3
	128	12	19.0	9.5	812	685	312.6	8.9	76.2	0.9702	97.7
Qwen2.5-0.5B	32	12	21.0	26.7	2 945	1585	53.8	8.9	223.7	0.9993	96.9
	64	12	20.9	30.6	3 033	1585	195.6	8.9	231.2	0.9994	98.4
	128	12	20.1	39.8	3 224	1585	743.6	8.9	248.0	0.9993	99.2
Qwen2.5-1.5B	64	12	21.0	63.1	10 819	1625	393.6	8.9	577.1	0.9998	98.4
	128	12	20.2	142.0	11 179	1625	1491.9	8.9	670.5	0.9998	99.2
Llama-3.2-1B	64	12	21.6	71.1	8 020	2209	292.6	8.9	502.3	0.9961	100.0
	128	12	21.2	126.3	8 288	2209	1122.0	8.9	568.3	0.9953	100.0

rounds. Because a single WAN round trip costs 50 ms, every round-paying row lands at 0.10–6.53 s per isolated call where the certified gate stays at 0.14 ms. Second, the advantage survives at model scale in the same threat model: composing BERT-Base, GPT-2 and Llama-3.2-1B on the generic active row costs 33–152× more WAN latency than the measured DUPLEX executions. Third, among dealer-free systems DUPLEX has the lowest per-model totals on every profile in Table 11 (9.3× under BumbleBee on BERT-Base LAN and 7.8× on WAN), while also holding the strictly stronger active threat model. The disadvantages are stated once: SIGMA's trusted-dealer GPU operating point stays ≈6× faster on a LAN on BERT-Base (≈8× on GPT-2), and dealer removal is paid in preprocessing volume and rounds (Table 12, Preprocessing columns; §8.2).

Table 11: End-to-end latency decomposition across network profiles (LAN 0.05 ms/23.5 Gbps, MAN 6 ms/1 Gbps, WAN 50 ms/200 Mbps). DUPLEX (dealer-free, actively secure, two-party) compute, transmission and rounds are measured. Baseline rows re-evaluate each system at its published operating point under our network profiles and with matched sequence lengths (^{G}: GPU host; the SPDZ-style row is composed from the measured per-call costs of Table 12 and is wavy-underlined). Totals are computed before rounding. Bold: best dealer-free total among rows of the same model.

		Compute	Rounds	LAN		MAN		WAN	
Method	**Model**	(s)		trans. (s)	total (s)	trans. (s)	total (s)	trans. (s)	total (s)
DUPLEX (ours)	BERT-Base	6.6	685	0.3	**6.9**	10.6	**17.2**	66.7	**73.3**
	GPT-2	5.5	661	0.3	**5.8**	9.8	**15.3**	62.3	**67.8**
	OPT-125M	5.5	685	0.3	5.8	10.0	15.5	63.9	69.4
	Qwen2.5-0.5B	23.2	1585	1.1	24.4	33.8	57.0	200.6	223.8
	Qwen2.5-1.5B	63.1	1625	3.8	66.9	96.3	159.4	514.0	577.1
	Llama-3.2-1B	71.1	2209	2.8	73.9	77.4	148.5	431.2	502.3
SIGMA^G (Gupta et al., 2024)	BERT-Base	0.9	≈470	0.4	1.2	10.7	11.6	63.1	64.0
	GPT-2	0.5	≈510	0.2	0.7	6.0	6.6	40.3	40.8
BumbleBee (Lu et al., 2025)	BERT-Base	60.6	$\sim 10^3$	4.0	64.6	98.4	159.0	512.0	572.6
	GPT-2	66.0	$\sim 10^3$	1.0	67.0	28.8	94.8	164.0	230.0
SPDZ-style 2PC (Damgård et al., 2012; Cramer et al., 2018)	BERT-Base	8 508	6 370	20.3	8 528	508.0	9 016	2 667	11 175

Table 12: One certified nonlinear activation across two-party baselines, and DUPLEX at model scale. Online columns are per call (per inference for the model rows); DUPLEX figures are the measured GELU gate (the SPDZ-style, garbled and OT baselines use the cheaper ReLU). LAN/WAN totals combine compute with computed transmission (LAN 0.05 ms/23.5 Gbps, WAN 50 ms/200 Mbps); entries marked with a wavy underline denote the unique set of properties achieved exclusively by this work (their improvement factors are lower bounds), and bold with a wavy underline marks the combination of properties attained only by this work. The shaded column is the WAN-total improvement of the DUPLEX gate over each row (from unrounded values); for the model rows it is the improvement over the same model composed on the SPDZ-style row.

			Online (per call)				**Preprocessing**		
System	**Dealer**	**Adversary**	Comm. (B)	Rounds	LAN total	WAN total	Comm.	Rounds	**Improv.**
Executed on two hosts (E)									
BGI DPF LUT (14-bit) (Wagh, 2022)	dealer	semi-honest	0	0	17.0 ms	17.0 ms	dealt (263 B key)		118×
SPDZ-style 2PC (Damgård et al., 2012; Cramer et al., 2018)	none	active	8288	130	35.3 ms	6.53 s	auth. triples/bits		45 190×
DUPLEX (ours)	**none**	**active**	**0**	**0**	**0.14 ms**	**0.14 ms**	**39.8 KB**	**230**	**1×**
On two hosts using our measured primitives (M)									
Garbled circuits (SH)	none	semi-honest	1024	2	0.10 ms	100.0 ms	garbling/COT		≥692×
Authenticated garbling	none	active	2048	2	0.10 ms	100.1 ms	garbling/COT		≥692×
OT-based mixed (mal.)	none	active	2340	7	0.35 ms	350.1 ms	garbling/COT		≥2 423×
Baselines re-executed and measured on our two hosts (B)									
SIGMA (Gupta et al., 2024)	dealer	semi-honest	58	7	0.35 ms	350.0 ms	dealt (1.43 KB key)		2 422×
MUSE (Lehmkuhl et al., 2021)	none	client-mal.	≈1 300	≈4	0.25 ms	200.1 ms	GC+CDS, ≈383 KB		1 385×
SIMC (Chandran et al., 2022)	none	client-mal.	11 900	≈4	0.21 ms	200.5 ms	none		1 388×
BumbleBee (Lu et al., 2025)	none	semi-honest	410	≈10	0.50 ms	500.0 ms	HE setup		3 461×
DUPLEX *end-to-end on models from Table 11 (per inference)*									
DUPLEX – BERT-Base	none	active	812 MB	685	6.9 s	73.3 s	**dealer-free**		152×
DUPLEX – GPT-2	none	active	732 MB	661	5.8 s	67.8 s	**dealer-free**		71×
DUPLEX – Llama-3.2-1B	none	active	8 020 MB	2 209	73.9 s	502.3 s	**dealer-free**		33×