UVCC Phase 6: Private, Verifiable, Fully-Parallel GPU Computation across Untrusted Domains

Alien Team

Abstract—We present UVCC (Universal Verifiable Confidential Computing), a system enabling private and verifiable GPU computation across mutually distrustful administrative domains. UVCC combines: (i) confidentiality via three-party replicated secret sharing (RSS), (ii) verifiability via a deterministic transcript-of-transcripts construction yielding subsession, replica, and global roots, and (iii) native ML parallelism (DP/PP/TP) in a C++ runtime integrating GPU kernels, a reliable exactly-once transport, and NCCL-based collectives. We report a complete Phase 6 bring-up, diagnose and eliminate a PP deadlock, and demonstrate scale-out to R=8, S=4, T=2, M=32 (192 workers) with a determinism proof by matching global roots across reruns. We provide auditor-grade artifacts and figures derived automatically from full logs.

Index Terms—Confidential computing, verifiable computation, secure multiparty computation, GPUs, NCCL, DP/PP/TP parallelism, transcripts, deterministic auditing.

I. Introduction

ML training and inference at scale rely on aggressive parallelism across GPUs. Achieving confidentiality and verifiability across mutually distrustful domains while preserving native parallel performance remains challenging. UVCC executes client programs using honest-majority MPC across three domains, with a deterministic transcript architecture that makes auditing practical.

a) Contributions.: (1) A native GPU runtime (Phase 6) implementing DP/PP/TP with NCCL, deterministic IDs, and exactly-once transport. (2) A transcript-of-transcripts design with domain-separated hashing, producing subsession, replica, and global roots. (3) A complete bring-up at realistic scale (R=8, S=4, T=2, M=32) with a determinism proof. (4) Auditor artifacts and figures derived from full logs.

II. Model and Preliminaries

A. System and Adversary Model

We have three parties (domains) operating GPUs and networks. The adversary statically corrupts at most one party (honest-majority). Channels to a lightweight relay are authenticated. Each party enforces binary integrity under operator policy.

B. Replicated Secret Sharing (RSS)

Definition 1 (Replicated Secret Sharing over $\mathbb{Z}_{2^{64}}$). Let R=$\mathbb{Z}_{2^{64}}$ with addition mod 2^{64}. A value $x \in R$ is shared as (x_{01}, x_{12}, x_{20}) with $x \equiv x_{01}+x_{12}+x_{20} \pmod{2^{64}}$, and party i holds two consecutive pieces (e.g., party 0 holds (x_{01}, x_{20})). Addition and scalar multiplication act locally; multiplication uses standard 3PC protocols with preprocessed correlations.

We use arithmetic shares in R and Boolean shared forms as needed. OPEN reveals authenticated linear combinations with transcript commitments.

C. Transcript Leaves and Roots

Each protocol event emits a leaf with deterministic key and payload digest.

Definition 2 (Transcript Leaf). For epoch e, subsession identifier u, stream s, op identifier o, and chunk index c, a leaf is a tuple $\ell = \big(\text{ver}, e, u, s, o, c, \text{kind}, \text{hdr}, H(\text{payload})\big)$, with a versioned domain separator and canonical serialization. The leaf key is (e, u, s, o, c).

Definition 3 (Epoch, Replica, Global Roots). Let L_e be leaves in epoch e, sorted lexicographically by key; $\rho_e = \text{MerkleRoot}(L_e)$. For replica r, let U_r be subsession roots combined in canonical order; the replica root is $R_r = H(\texttt{UVCC\_REPLICA\_ROOT\_V1} \,\|\, \texttt{sid\_rep} \,\|\, e \,\|\, U_r)$. The global root is $G = H(\texttt{UVCC\_GLOBAL\_ROOT\_V1} \,\|\, \texttt{sid\_job} \,\|\, e \,\|\, R_0 \| R_1 \| \dots)$.

D. IDs and Deterministic Derivations

UVCC derives IDs to prevent collisions across DP/PP/TP:

$\texttt{msg\_id32} := \text{head}_4\big(H(\texttt{sid} \,\|\, \texttt{sgir\_op\_id32} \,\|\, \texttt{src} \,\|\, \texttt{dst} \,\|\, \texttt{chunk\_}$

$\texttt{stream\_id64} := F(\texttt{sid}, \texttt{op\_ids}), \qquad \texttt{fss\_id} := F'(\texttt{sid}, \texttt{comp\_id}, \texttt{c}$

with F, F' domain-separated PRGs. DP/PP/TP subgroups use distinct `sid_sub` to avoid reusing PRG streams or stream IDs.

E. OPEN and MUL Protocols

a) OPEN.: Parties reveal authenticated linear combinations by exchanging the appropriate share components and reconstructing $x = x_0+x_1+x_2$:

```
# Party P_i holds (x_i, x_{i+1}) over Z_2^64
OPEN_i(x_i, x_{i+1}):
  send_to(P_{i+1}, x_i); recv x_{i-1} from P_{i-1}
  x = x_{i-1} + x_i + x_{i+1}
  return x
```

b) Beaver Multiplication (matrix/elementwise).: Using preprocessed (a, b, c) with $c = a \cdot b$, parties compute $E = X-A$, $F = Y-B$, OPEN(E, F), then locally combine to obtain shares of $Z = XY$; batching merges (E, F) OPENs.

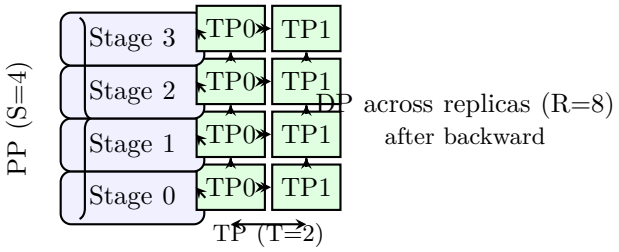

Fig. 1. Native parallelism topology.

F. DP/PP/TP Semantics over RSS

a) SR-DP.: The aggregate gradient is $\sum_{r=0}^{R-1} g^{(r)}$, realized as a local allreduce per party; no cross-party MPC is required for DP reduction.

b) TP.: Per-stage collectives (allreduce/allgather) are linear on shares and preserve secrecy.

c) PP.: Staged microbatches (1F1B) interleave compute and communication; the scheduler keeps sufficient in-flight microbatches to hide OPEN latency while preserving deterministic local ordering within each (r, s, t) subgroup.

III. Design Overview

A. Transport

We implement exactly-once frame acceptance with canonical encoding, CRC32C, idempotent application keyed by message IDs, conservative retransmission, and long-polling. This tolerates domain skew and reduces overload on the relay.

Proposition 1 (Exactly-once Consumption). If the relay provides at-least-once delivery with acknowledgments and recipients perform idempotent acceptance keyed by message IDs, then application-level processing is exactly-once.

Retransmission uses capped backoff with generous retry budgets to avoid aborting before higher-level deadlines; duplicate frames must match the committed hash or are rejected.

B. Native Parallelism (DP/PP/TP)

UVCC constructs TP, PP, and DP groups deterministically and initializes NCCL communicators with robust UID distribution and deadlines derived from a configurable Phase 6 timeout.

IV. Formal Properties

A. Determinism

Theorem 1 (Deterministic Roots). Fix `sid_job`, topology (R, S, T, M), and seeds for ID derivation. If (i) leaf keys are unique and (ii) each group operation is deterministic given inputs, then subsession, replica, and global roots are deterministic functions of inputs and topology.

Proof sketch. Determinism follows by composition: canonical key ordering ensures a unique Merkle root per epoch; replica/global roots are deterministic domain-separated hashes of ordered child roots. With fixed seeds and message ordering constraints, the runtime emits a unique sequence of leaves. Hence all derived roots are unique. □

B. Confidentiality

Lemma 1 (One-corruption Privacy for Linear Operations). Under RSS over $R = \mathbb{Z}_{2^{64}}$, addition and scalar multiplication are information-theoretically private against one corrupted party.

Proof sketch. Each party sees two random shares whose sum (mod 2^{64}) is uniform conditioned on the secret; linearity preserves indistinguishability. Standard arguments apply. □

Theorem 2 (Confidentiality). Assuming standard 3PC multiplication with correlations and authenticated OPEN, UVCC maintains confidentiality of inputs, model, optimizer state, gradients, and intermediate values against one corrupted domain.

C. Verifiability and Proof Bundle

Let $\mathcal{B}$ be the bundle containing epoch roots, subsession roots, replica roots, the global root, and attested identities/policy commitments. A verifier recomputes Merkle roots and domain-separated hashes from $\mathcal{B}$ and checks signature bindings. Optionally, sampled kernel checks (SKS) bind algebraic tests (e.g., Freivalds) into transcript leaves for higher assurance.

V. Implementation

The C++ runtime integrates: (i) a reliable transport with exactly-once acceptance and long-polling, (ii) a transcript store with deterministic keys and Merkle roots, (iii) OPEN/LIFT engines with batch TLV framing, and (iv) NCCL-based groups for TP/PP/DP with robust UID exchange and deadlines based on Phase 6 timeout.

a) Scheduler Correctness and Deadlock Avoidance.: PP gradient receives are posted per microbatch after forward completes on the stream to avoid single-stream backpressure that could deadlock forward sends; DP initialization is staged after backward to avoid OPEN skew during NCCL bootstrap.

VI. Evaluation

A. Setup

Final run: R=8, S=4, T=2, M=32 on 24 pods across providers. We release a consolidated, redaction-safe log file with runner, all 192 workers, and the audit bundle. Metrics and figures below are auto-derived from those logs.

B. NCCL Initialization

C. Protocol Coverage and Robustness

D. Determinism Proof

Two full runs with fixed `sid_job` yield identical `global_root_hex` values, demonstrating determinism under cross-provider skew.

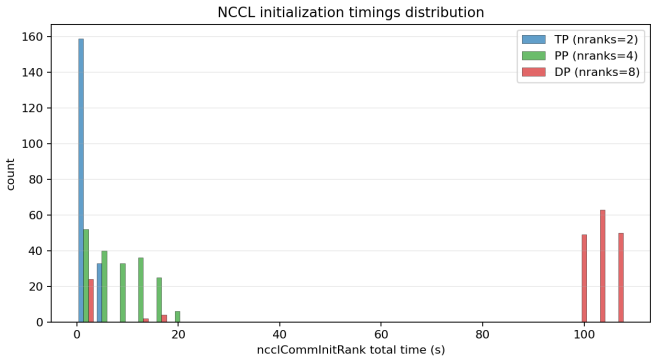

Fig. 2. NCCL init timings by nranks (TP=2, PP=4, DP=8).

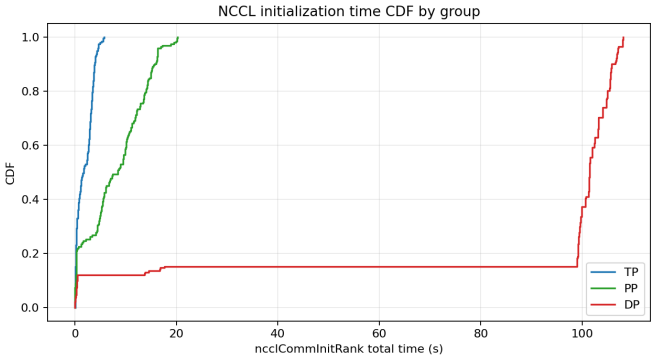

Fig. 3. NCCL init time CDF by group (TP/PP/DP).

VII. Threats to Validity and Ethics

Deployment variance (provider scheduling, NICs) may affect timings but not determinism or correctness. Logs are redaction-safe; no client secrets are present. We do not place transcripts on-chain; only hash commitments are published as needed for audit.

VIII. Related Work

We relate to honest-majority MPC over rings, verifiable computation with transcripts, and large-scale ML parallelism (DP/PP/TP/NCCL). UVCC's novelty is combining auditor-grade verifiability with native GPU parallelism across domains at realistic scale.

IX. Conclusion

UVCC Phase 6 shows private, verifiable, fully-parallel GPU computation across untrusted domains is practical and deterministic. Future work: GPU-accelerated preprocessing (TCF/W-VOLE), production SKS, and larger models.

Appendix

We summarize the confidentiality game and leakage model, aligning with research/privacy_new.txt. Leakage is limited to message sizes and deterministic scheduling; secret values remain hidden under RSS.

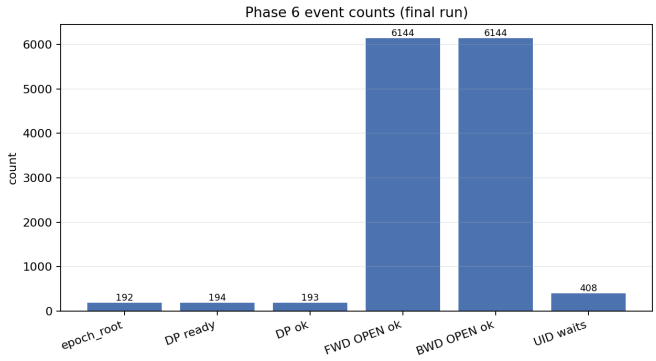

Fig. 4. Event counts: epoch roots, OPEN completions, DP readiness.

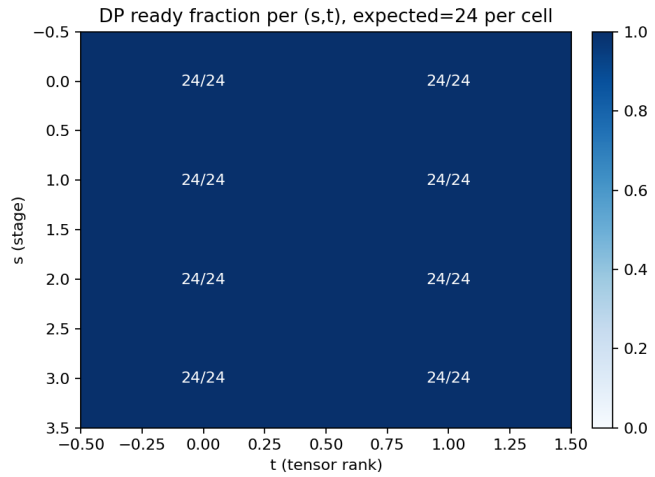

Fig. 5. DP readiness fraction per (stage s, tensor t), annotated as ready/expected. Uniform color indicates all DP groups reached readiness.

We overview stream/ID allocation and scheduling tables from research/PARALLEL.txt, including ordering constraints and deterministic ID derivations for TP/PP/DP.

All figures are generated from the consolidated log by research/uvcc_native/scripts/make_phase6_figs.py. The output directory contains runner logs, 192 worker logs, per-worker roots, and the audit bundle with global root.

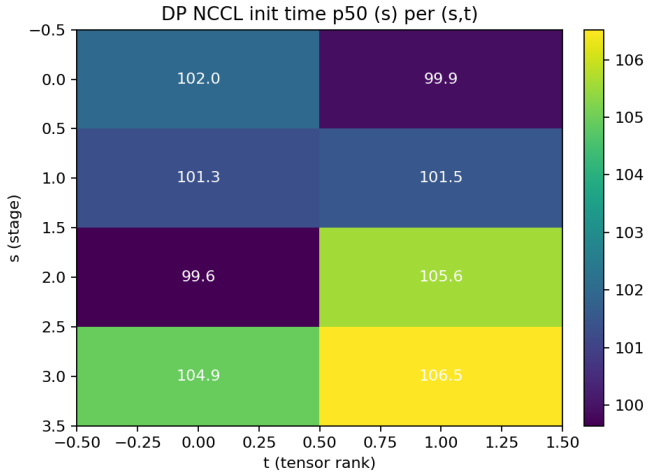

Fig. 6. DP NCCL initialization latency p50 per (stage s, tensor t); highlights cross-pod skew even when readiness is complete.

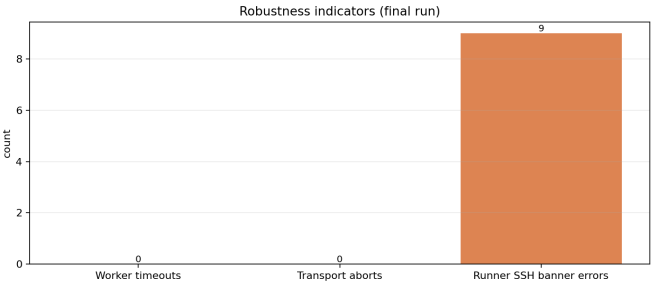

Fig. 7. Robustness indicators (final run): worker timeouts and transport aborts are zero; runner SSH banner errors occurred during artifact collection and were tolerated.

TABLE I
Key Phase 6 metrics (derived from consolidated logs)

Metric	Count/Value
epoch root	192
dp ready	194
dp ok	193
fwd open ok	6144
bwd open ok	6144
uid wait	408
worker timeouts	0
transport aborts	0
runner ssh banner errors	9
nccl init tp (n=192) min/p50/max	0.120/1.710/5.750 s
nccl init pp (n=192) min/p50/max	0.010/8.450/20.280 s
nccl init dp (n=192) min/p50/max	0.020/101.510/108.060 s

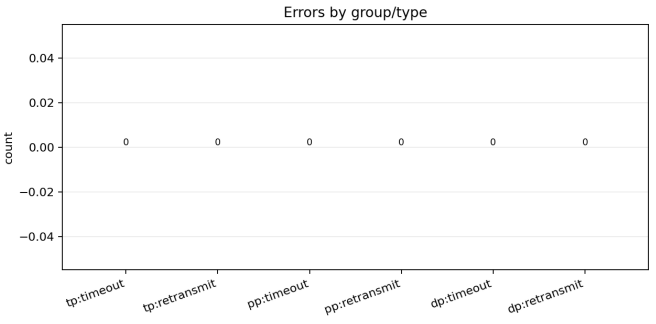

Fig. 8. Errors by group/type (TP/PP/DP).