TEE-Assisted Authenticated MPC for Resource-Constrained Edge Intelligence

Puyu Wang, Ruidan Su, *Member, IEEE*, Zhenshen Liu, Ruiqi Yang, Kai Fan, *Member, IEEE*, Hui Li, *Member, IEEE*,

***Abstract*—Supporting privacy-preserving analytics across large Internet of Things (IoT) populations remains difficult. Resource-constrained devices may be unable to execute expensive multi-party computation (MPC) preprocessing or remain in a sustained many-party online protocol. We present a remotely attested, trusted execution environment (TEE)-assisted authenticated MPC multiplication scheme for large-scale edge intelligence. The TEE is deliberately restricted to the offline phase: before task-time compute committees are selected according to current edge availability and resource conditions, an Intel Software Guard Extensions (SGX) enclave provisions fine-grained authenticated material to authorized edge devices and then leaves the computation path. When a task arrives, a smaller set of $m < n$ compute committees aggregates the delegated device shares, the input owners inject their private data and model parameters through preprocessed masks, and the committees perform authenticated online multiplication without re-entering the TEE. We establish the correctness and authentication invariants of the committee-level computation and analyze the remotely attested deployment under an explicit TEE trust assumption. A hardware-SGX prototype implements the integrated scheme. The prototype generates batches of 10 million triples; at $l = 1$ million in the evaluated three-recipient configuration, it achieves a $12.17\times$ preprocessing speedup over the evaluated MP-SPDZ MASCOT baseline. A separate stress test packages one preprocessing row for 100,000 recipients, and integrated tests reject ciphertext and online message-authentication-code (MAC) tampering. These results show that our scheme substantially improves preprocessing performance over the evaluated software-only protocol and is well suited to resource-constrained edge-intelligence deployments, underscoring its strong practical utility.**

***Index Terms*—Internet of Things; Multi-Party Computation; Trusted Execution Environment; Edge Intelligence.**

I. Introduction

LARGE-SCALE IoT deployments increasingly require privacy-preserving inference and analytics across independently managed cameras, sensors, meters, and industrial endpoints [1]. Secure multi-party computation (MPC) allows mutually distrustful parties to evaluate an agreed function without pooling raw measurements, but deploying maliciously secure MPC at the edge remains difficult because endpoints have limited computation, memory, energy, and connectivity [2].

Practical arithmetic MPC commonly combines additive secret sharing with Beaver multiplication triples [3], [4]. Additions and multiplications by public constants are local, whereas every secret–secret product consumes input-independent correlated randomness. The offline phase must therefore generate and securely distribute large quantities of authenticated triples and input-recovery masks, which often dominate resource consumption in multiplication-intensive edge MPC. This demand is particularly acute in private edge inference, where an edge owner protects an input X and a model provider protects weights W. Each product $x \cdot w$ evaluated after secret sharing consumes one authenticated triple, so dense and convolutional layers require large triple batches.

Currently, software-only preprocessing remains costly: MASCOT uses oblivious-transfer extension [5], LowGear and HighGear use homomorphic encryption [6], and pseudorandom correlation generators expand short seeds locally [7]–[9]. Although these approaches avoid a centralized triple generator, they retain substantial cryptographic computation, interaction, or engineering complexity for constrained devices [10]. In contrast, Intel SGX provides hardware-isolated enclaves whose code and data are protected from privileged software on an otherwise untrusted host, while Data Center Attestation Primitives (DCAP) allows remote recipients to verify the enclave measurement and approved code before accepting its output [11]–[13]. Together, these mechanisms allow an attested enclave to assume the preprocessing-dealer role without requiring devices to trust the host operator. Generating triples and MAC material inside the enclave and distributing device-specific shares over encrypted channels frees resource-constrained devices from OT/HE-based interactive preprocessing, substantially reducing their computation and communication burden. This benefit comes with an explicit hardware trust assumption: the enclave observes the triples and global MAC key, but receives no application input or model parameter and is released before the online phase.

Edge devices often cannot sustain either expensive preprocessing or repeated many-party online communication. We therefore propose the TEE-assisted offline-to-online architecture shown in Fig. 1, which separates *offline share ownership* from *online compute topology*. During preprocessing, the authorized n edge devices acting as data providers are known, but the task-specific committees are not. The enclave provisions fine-grained authenticated shares to these devices and then exits the computation path. When a task arrives, the system selects $m < n$ compute committees according to current availability, compute and memory capacity, network conditions, data locality, and load. Throughout this paper, a *compute committee* C_j denotes one logical MPC participant that holds one aggregate value share, MAC share, and MAC-

Corresponding author information and funding acknowledgments will be added before submission.

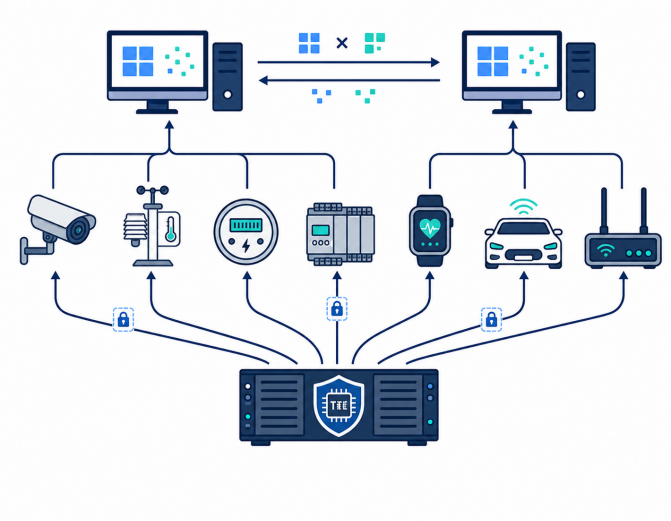


Fig. 1. Overview of the proposed TEE-assisted edge MPC architecture.

key share. It may be implemented by a single edge gateway or by a service cluster whose internal nodes jointly realize that participant. This late binding accommodates the time-varying edge environment and uncertain, deployment-specific upstream connectivity without fixing the online topology in advance or re-entering the TEE. Each device delegates the selected unused shares to exactly one committee; the committees aggregate them, the authorized input owners inject their secrets through preprocessing masks, and the committees execute Beaver multiplication and MACCheck. After delegation and input injection, the edge devices need not remain in the sustained interactive computation.

This paper makes five contributions:

• **Device-First Dynamic-Committee Architecture:** We decouple the stable set of n preprocessing recipients from task-time selection of $m < n$ compute committees. This late binding adapts to changing edge conditions while keeping application inputs outside the TEE and freeing the TEE during the online phase.

• **Attested Authenticated Preprocessing and Secure Distribution:** Our SGX service generates shared epoch-level SPDZ MAC material, authenticated Beaver triples, and independent input masks. Signed manifests, recipient-specific ECDH/HKDF session keys, AES-GCM streaming, and DCAP protect and bind each authorized batch.

• **Secure Transition to Committee-Level Authenticated MPC:** Authenticated assignments and tuple identifiers bind one-time device delegation to a task and committee partition. Linear aggregation preserves the represented values, Beaver relation, global MAC-key sum, and MAC invariant, enabling masked input injection, Beaver multiplication, and batched MACCheck without re-entering the TEE.

• **Explicit Security and Lifecycle Analysis:** We formalize authenticated preprocessing in the $\mathcal{F}_{\mathsf{TEE}}$-hybrid model and give a simulation-based security proof framework with abort against a static malicious adversary corrupting any proper subset of recipients. We also establish the committee-aggregation invariants and analyze binding, integrity, replay, one-time use, corruption, crash recovery, and device unavailability.

• **Genuine-SGX Implementation and Evaluation:** Our hardware-SGX prototype integrates DCAP, bounded-buffer streaming, independent recipient verification, and committee execution. Separate stress tests evaluate $(n, l) = (100{,}000, 1)$ and $(3, 10^7)$, rather than both maxima simultaneously, and demonstrate substantial preprocessing speedups, while integrated tests validate the complete computation and tamper-rejection path.

Section II reviews the background, Section III presents the protocol, Section IV analyzes its security, Section V evaluates the prototype, and Section VI concludes.

II. Preliminaries

A. Related Work

Privacy-preserving IoT and edge-intelligence systems combine cryptographic protocols, trusted execution, and distributed learning to balance data utility, privacy, and endpoint constraints [2], [10]. Arithmetic MPC is attractive because it protects both data and intermediate values, but its preprocessing can dominate resource use when a circuit contains many secret multiplications.

MASCOT generates authenticated triples via oblivious transfer (OT) extension [5], whereas LowGear and HighGear use homomorphic encryption (HE) [6]; pseudorandom correlation generators reduce communication through local expansion from short seeds [7]–[9]. These software approaches avoid centralized trust but retain substantial cryptographic computation or interaction, while TEE-assisted preprocessing instead trades stronger hardware trust for removing OT/HE from constrained recipients; our enclave handles neither application inputs nor model parameters. Prior work covers outsourced conversion of SPDZ-style authenticated shares [14], preprocessing for dynamic subsets and committee switching [15], and committee techniques for large-scale MPC [16]. In our implementation, an attested TEE provisions fine-grained shares to n devices before the online topology is fixed, after which a one-time task assignment forms $m < n$ committees that perform masked input recovery, Beaver multiplication, and batched MACCheck without reconstructing triples or re-entering the TEE.

B. Software Guard Extensions (SGX)

Intel SGX provides hardware-isolated enclaves whose code and data are protected from privileged host software [11], [12]. Our scheme uses this isolation only during offline preprocessing, where the enclave generates authenticated triples, MAC material, and device-specific ciphertexts; application inputs and model parameters remain outside the enclave.

SGX DCAP provides Elliptic Curve Digital Signature Algorithm (ECDSA)-based remote attestation [13]. An enclave first invokes EREPORT to create a local *Report* containing its code measurement, security attributes, and a 64-byte user-defined report_data field protected by a hardware MAC. The platform's Quoting Enclave verifies this Report and produces a signed *Quote* that carries the enclave identity and report_data together with certification evidence. A recipient verifies the

Quote, expected measurement, attestation collateral, and platform trusted computing base (TCB) status before trusting the enclave.

In our protocol, report_data directly binds the enclave's volatile signing public key g_{pub} and the batch context to the attested enclave. Each recipient-specific ephemeral elliptic-curve Diffie–Hellman (ECDH) public key pk_{E_i} is authenticated indirectly by a KEYENC signature under the corresponding volatile private key g_{priv}, rather than being placed directly in report_data. A verified recipient first authenticates g_{pub} through the Quote and then uses it to verify the ECDH key before establishing the encrypted channel. Remote attestation thus authenticates the preprocessing source and channel.

C. Authenticated Secret Sharing

Our protocol uses authenticated additive sharing over $\mathbb{F}_p$, where $p = 2^{127} - 1$. For a secret x, device D_i holds a value share x_i, a global-MAC-key share α_i, and a MAC share $\gamma_i(x)$ satisfying

$$x = \sum_{i=1}^{n} x_i, \qquad \alpha = \sum_{i=1}^{n} \alpha_i, \qquad \alpha x = \sum_{i=1}^{n} \gamma_i(x) \pmod{p}.$$

We denote the authenticated sharing by $\langle x \rangle = ([x], [\gamma(x)])$. Opening reconstructs x while keeping α secret shared; a batched MACCheck combines the local residual shares under public random coefficients and accepts only when their aggregate is zero.

D. Multiplication Triples

Beaver triples enable multiplication over additive secret sharing via preprocessing. In our field instantiation, a triple consists of three shared random values $([a], [b], [c])$ such that $c = a \cdot b$ over $\mathbb{F}_p$. Triples are independent of inputs and can be generated in the offline phase. Their authenticated online use in the proposed committee topology is specified in Section III.

III. Proposed TEE-Assisted Scheme

A. Scheme Architecture and End-to-End Workflow

The scheme has three phases. Phase 1 follows the standard SGX/DCAP provisioning path summarized in the Appendix: the TEE verifies a signed manifest for an ordered set of n authorized edge data providers, establishes attested device-specific sessions, generates authenticated triples and input masks, encrypts and delivers each device's local shares, and leaves the computation after recipient verification. The manifest fixes the recipient set and MAC epoch but not the online topology. In Phase 2, an authenticated task assignment selects $m < n$ committees and partitions the device indices; each required source share is delegated exactly once and aggregated before the authorized data and model owners inject their private inputs through the masks. In Phase 3, the committees alone perform Beaver multiplication and MACCheck, while the TEE and original edge recipients no longer participate in the sustained arithmetic.

B. Initialization and Recipient Authorization Phase

Before an enclave invocation, a deployment authority creates a signed recipient manifest

$$man = (v, n, l, \mathsf{prefix}, t_{\mathsf{iss}}, t_{\mathsf{exp}}, \mathsf{epoch}, \mathsf{challenge}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n, \sigma_A).$$

The authority generates σ_A by signing the canonical binary encoding of all preceding manifest fields with its ECDSA-P256 private key. The signature enables the enclave and recipient devices to verify the manifest's origin and detect unauthorized modification. Here n is the number of edge-device recipients. The corresponding verification key pk_A is compiled into the enclave and therefore covered by the enclave measurement (MRENCLAVE). The enclave verifies the signature, version, batch parameters, and ordered P-256 recipient keys. It computes $h_{\mathsf{man}} = \mathsf{SHA256}(man)$ using the SHA-256 cryptographic hash function to identify the authorized manifest and computes h_{members} over the ordered recipient-key list to identify the device set. These digests bind subsequent session establishment, attestation, and epoch state to the same manifest and recipients. Each device independently verifies the manifest signature and validity and confirms that its public key appears at its assigned index before accepting its output. The mathematical notation remains one-based, $i \in \{1, \ldots, n\}$, whereas the wire format and Hash-based Message Authentication Code (HMAC)-based Extract-and-Expand Key Derivation Function (HKDF) context use the zero-based party index $\mathsf{idx}_i = i - 1 \in \{0, \ldots, n-1\}$.

The 256-bit epoch identifies the lifetime of the SPDZ MAC key for one fixed ordered recipient set. On first use, the enclave samples an epoch seed, binds it to $(\mathsf{epoch}, h_{\mathsf{members}})$, and seals it. Reusing the same epoch with a different ordered recipient set fails closed. The committee partition is intentionally absent from the offline manifest and may be selected only after the devices have received their shares. Before any tuple in an epoch or assigned tuple range is delegated, the devices bind that material to one authenticated task assignment. Repartitioning material protected by the same MAC key requires a fresh epoch or a secure MAC-share refresh. A fresh 256-bit challenge is included in every manifest and is subsequently bound to the Quote context and recipient transcript.

C. Secure Session Establishment Phase

Algorithm 1 establishes an authenticated session key with each recipient. Here, $\mathsf{label}_{\mathsf{key}}$ identifies the key-derivation process, $\mathsf{LE}(P)$ is the 64-byte SGX encoding $P_x \parallel P_y$ of a P-256 point, and $\mathsf{BE64}(\mathsf{nonce}_k)$ is the eight-byte big-endian encoding of the invocation nonce. The subsequent DCAP Quote binds the volatile signing key g_{pub} to the enclave measurement and authorized batch context.

D. Authenticated Preprocessing Provisioning Phase

After Π_{KeyGen}, Π_{TriGen} operates over $\mathbb{F}_{2^{127}-1}$. A sealed epoch seed derives a nonzero α and shares $\{\alpha_i\}$ bound to $(\mathsf{epoch}, h_{\mathsf{members}})$. Algorithm 2 then samples each triple $(a^t, b^t, c^t = a^t b^t)$ and masks r_x^t, r_y^t, and independently shares

Algorithm 1 Enclave Secure Session Establishment and Key Encapsulation (Π_{KeyGen})

Input: Verified manifest man, recipient keys $\{\mathsf{pk}_i\}_{i=1}^{n}$, manifest hash h_{man}, and current invocation nonce nonce_k.
Initialization:

$$g_{\mathsf{priv}} \leftarrow \{0,1\}^{256}, \quad g_{\mathsf{pub}} \leftarrow \mathsf{GenPub}(g_{\mathsf{priv}})$$

Key Encapsulation for Each Recipient Device D_i:
for $i \leftarrow 1$ **to** n

$$\mathsf{sk}_{E_i} \leftarrow \{0,1\}^{256}, \quad \mathsf{pk}_{E_i} \leftarrow \mathsf{GenPub}(\mathsf{sk}_{E_i})$$
$$\mathsf{info}_i \leftarrow \mathsf{label}_{\mathsf{key}} \,\|\, \mathsf{Hex}(h_{\mathsf{man}}) \,\|\, \texttt{':'} \,\|\, \mathsf{ASCII}_{10}(\mathsf{idx}_i)$$
$$\mathsf{secret}_i \leftarrow \mathsf{ECDH}(\mathsf{sk}_{E_i}, \mathsf{pk}_i), \qquad \mathsf{salt}_i \leftarrow \{0,1\}^{256}$$
$$K_i \leftarrow \mathsf{HKDF\text{-}SHA256}_{128}(\mathsf{secret}_i, \mathsf{salt}_i, \mathsf{info}_i)$$
$$M_i \leftarrow (\mathsf{LE}(\mathsf{pk}_{E_i}) \,\|\, \mathsf{info}_i \,\|\, \mathsf{salt}_i \,\|\, \mathsf{BE64}(\mathsf{nonce}_k))$$
$$\sigma_i \leftarrow \mathsf{ECDSA}_{g_{\mathsf{priv}}}(M_i)$$
$$\mathsf{KEYENC}_i \leftarrow (\mathsf{LE}(\mathsf{pk}_{E_i}), \mathsf{salt}_i, \mathsf{BE64}(\mathsf{nonce}_k), \sigma_i)$$

Output: The enclave retains $\{K_i\}_{i=1}^{n}$ only for the current invocation and sends KEYENC_i to D_i.

every value and MAC $\gamma(\xi) = \alpha\xi$. Each recipient row contains the five value shares and their five MAC shares.

BATCH_i canonically binds the manifest, epoch, recipient, key, and batch parameters, while schema is the newline-terminated ASCII field-order declaration `a b c rx ry mac_a mac_b mac_c mac_rx mac_ry`. Here SP denotes one ASCII space byte and LF denotes one line-feed byte.

The exact record is `alpha_share␣<decimal>\n`. The ALPHA_i record and l rows form the canonical transcript Q_i. The notation $\mathsf{Enc}_{K_i}^{\mathsf{AES\text{-}GCM}}$ denotes AES-GCM-128 streaming encryption under K_i and produces CT_i as an ordered sequence of individually authenticated frames rather than one monolithic ciphertext. In the implementation, ECDSA-SHA256 hashes its input before signing; therefore, the digest signed for the transcript is $\mathsf{SHA256}(H_i) = \mathsf{SHA256}(\mathsf{SHA256}(Q_i))$. The signed manifest, KEYENC, attested context, and transcript signature bind the recipient and batch context.

E. Remote Attestation Binding

The enclave maintains a separate sealed counter chain for each ordered membership digest h_{members}. A previously unseen membership starts with committed value $\mathsf{ctr}_0 = 0^{256}$; this is not a single global counter shared by all memberships. Each membership-scoped state contains a committed counter and, during an unfinished invocation, a pending record bound to the authorized manifest. After manifest authorization, if no pending record exists, the enclave samples the 64-bit nonce nonce_k and computes the candidate

$$\mathsf{ctr}_k = \mathsf{SHA256}(\mathsf{ctr}_{k-1} \,\|\, \mathsf{LE64}(\mathsf{nonce}_k)),$$

where ctr_{k-1} is that membership's committed value and $\mathsf{LE64}(\mathsf{nonce}_k)$ is the fixed-width eight-byte little-endian encoding of nonce_k. The enclave persists ctr_k as pending before output generation but does not yet advance the committed counter. Because the triple-generation randomness, g_{priv}, and the recipient-specific ephemeral ECDH keys are volatile, this pending record alone is insufficient to regenerate an interrupted batch. Recovery therefore follows one of two paths. If the complete artifacts from the interrupted invocation remain durably available, the host may replay the byte-identical ciphertexts and their associated signed and attested metadata under the original bid_k. Otherwise, the control plane abandons that batch, never activates it through a task assignment, resolves its pending state, and starts a new invocation with fresh cryptographic material, a new counter transition, and a distinct batch identifier. Regenerating any preprocessing material under an existing bid_k, or mixing artifacts from different attempts under that identifier, is prohibited.

The fixed strings d_{epoch} and d_{quote} distinguish the epoch-commitment hash from the Quote-context hash. Using the

Algorithm 2 Global MAC Key and Authenticated Triples Generation (Π_{TriGen})

Input: Epoch state, verified ordered recipient set, session keys $\{K_i\}$, signing key $(g_{\mathsf{priv}}, g_{\mathsf{pub}})$, and triple count l.
Epoch MAC Key:

$$(\alpha, \alpha_1, \ldots, \alpha_n) \leftarrow \mathsf{DeriveEpoch}(\mathsf{seed}_{\mathsf{epoch}}, h_{\mathsf{members}})$$
$$\alpha = \sum_{i=1}^{n} \alpha_i \pmod p$$

Authenticated Triples Generation:
for $t \leftarrow 1$ **to** l

$$a^t, b^t \leftarrow \mathbb{F}_p, \qquad c^t = a^t b^t$$
$$r_x^t, r_y^t \leftarrow \mathbb{F}_p$$
$$\xi = \sum_{i=1}^{n} \xi_i \pmod p$$

where $\xi \in \{a^t, b^t, c^t, r_x^t, r_y^t\}$.

$$\sum_{i=1}^{n} \gamma_i(\xi) = \alpha \cdot \sum_{i=1}^{n} \xi_i \pmod p$$
$$\langle \xi \rangle_i = (\xi_i, \gamma_i(\xi))$$
$$T_i^t = (\langle a^t \rangle_i, \langle b^t \rangle_i, \langle c^t \rangle_i, \langle r_x^t \rangle_i, \langle r_y^t \rangle_i)$$

Encryption and Transcript Authentication:

$$\mathsf{ALPHA}_i = \texttt{alpha_share} \,\|\, \mathsf{SP} \,\|\, \mathsf{ASCII}_{10}(\alpha_i) \,\|\, \mathsf{LF}$$
$$Q_i = \mathsf{BATCH}_i \,\|\, \mathsf{schema} \,\|\, \mathsf{ALPHA}_i \,\|\, T_i^1 \,\|\, \cdots \,\|\, T_i^l$$
$$\mathsf{CT}_i = \mathsf{Enc}_{K_i}^{\mathsf{AES\text{-}GCM}}(\mathsf{ALPHA}_i, T_i^1, \ldots, T_i^l)$$
$$H_i = \mathsf{SHA256}(Q_i), \qquad \sigma_i^{\mathsf{data}} = \mathsf{ECDSA\text{-}SHA256}_{g_{\mathsf{priv}}}(H_i)$$

Output: The untrusted host persists $\{\mathsf{CT}_i\}$ and public metadata; the enclave releases no plaintext share or session key.

pending counter, the enclave computes

$$\mathsf{com}_{\mathsf{epoch}} = \mathsf{SHA256}(d_{\mathsf{epoch}} \parallel \mathsf{epoch} \parallel h_{\mathsf{members}} \parallel \mathsf{seed}_{\mathsf{epoch}}).$$

Let $H = \mathsf{SHA256}$. The unique identifier of provisioning invocation k is

$$\mathsf{bid}_k = H(\texttt{'beaver-bid-v1'} \parallel h_{\mathsf{man}} \parallel \mathsf{ctr}_k \parallel \mathsf{com}_{\mathsf{epoch}}).$$

It then sets

$$\mathsf{ctx}_k = \mathsf{SHA256}(d_{\mathsf{quote}} \parallel h_{\mathsf{man}} \parallel \mathsf{ctr}_k \parallel \mathsf{com}_{\mathsf{epoch}}).$$

The 64-byte SGX report-data field is exactly

$$\mathsf{report}_k = H\big(\mathsf{LE}(g_{\mathsf{pub}})\big) \parallel \mathsf{ctx}_k$$

The local quoting infrastructure generates $\mathsf{Quote}_k \leftarrow \mathsf{DCAP.QuoteGen}(\mathsf{report}_k)$. Thus the Quote binds the enclave signing key to the signed manifest, ordered-recipient-set epoch, and pending counter state rather than to the signing key alone. Only after all recipient files have been published successfully does an explicit finalization transition ctr_k from pending to committed. Following a failed or interrupted publication, the system either republishes the complete byte-identical artifact set or abandons the old batch and creates a new bid_k; it never regenerates a different artifact set under the old identifier.

F. Recipient Verification and Decryption Phase

Data Packet Structure:

The recipient file is an ordered sequence

$$\mathcal{M}_i = (\mathsf{BATCH}_i, \mathsf{CTRCHECK}_k, \mathsf{KEYENC}_i, \mathsf{CT}_i, \mathsf{ATT}_i).$$

BATCH_i is the public batch header and encodes the manifest hash, epoch, $\mathsf{com}_{\mathsf{epoch}}$, challenge, recipient count, triple count, zero-based recipient index idx_i, prefix, and recipient public key. $\mathsf{CTRCHECK}_k$ carries $(\mathsf{nonce}_k, \mathsf{ctr}_k)$. Together, the verified manifest hash, counter, and epoch commitment determine bid_k. KEYENC_i has the format defined in the session-establishment phase, CT_i is the ordered frame sequence output by $\mathsf{Enc}_{K_i}^{\mathsf{AES\text{-}GCM}}$, and ATT_i contains $(g_{\mathsf{pub}}, \mathsf{ctx}_k, \sigma_i^{\mathsf{data}}, \mathsf{Quote}_k)$.

Session Key Decapsulation:

Device D_i first verifies the authority signature and validity interval of man, checks that its private key matches manifest entry i, and requires every BATCH_i field to match the manifest and local zero-based recipient index $\mathsf{idx}_i = i - 1$. It reconstructs info_i from $(h_{\mathsf{man}}, \mathsf{idx}_i)$, verifies the exact KEYENC_i signature, validates the received P-256 point, and derives

$$K_i = \mathsf{HKDF\text{-}SHA256}_{128}(\mathsf{ECDH}(\mathsf{sk}_i, \mathsf{pk}_{E_i}), \mathsf{salt}_i, \mathsf{info}_i).$$

TEE Remote Attestation Verification:

The receiver invokes the DCAP quote-verification library with current, unexpired collateral; rejects application programming interface (API) errors, invalid signatures, revocation, unknown terminal results, and debug enclaves unless explicitly authorized; and pins MRENCLAVE or the signer measurement (MRSIGNER) while enforcing a minimum independent-software-vendor security version number (ISVSVN) when configured. It recomputes bid_k and ctx_k and requires the Quote report data to equal $H\big(\mathsf{LE}(g_{\mathsf{pub}})\big) \parallel \mathsf{ctx}_k$.

Payload Decryption and Integrity Check:

The receiver decrypts every frame under K_i, rejects any GCM-tag failure, parses exactly one in-range α_i and exactly l ten-field rows, and reconstructs the canonical plaintext transcript Q_i defined in Algorithm 2. It sets $H_i = \mathsf{SHA256}(Q_i)$ and verifies σ_i^{data} using $\mathsf{ECDSA\text{-}SHA256}$ on H_i, including the API's second SHA-256 application. Frame deletion, duplication, reordering, or cross-recipient substitution therefore changes the row count or signed transcript even though the current authenticated-encryption-with-associated-data (AEAD) frames themselves carry no additional authenticated data (AAD).

Before releasing plaintext, the receiver checks $\mathsf{ctr}_k = \mathsf{SHA256}(\mathsf{ctr}_{k-1} \parallel \mathsf{LE64}(\mathsf{nonce}_k))$ against its locally stored checkpoint for the ordered membership. After all checks succeed, it labels the decrypted output with bid_k. The receiver uses a durable transaction journal to commit the plaintext output and checkpoint as a crash-recoverable two-file transaction. This detects replay for a stateful receiver without assuming an indivisible filesystem update of two files.

G. Task-Time Committee Setup and Input Sharing

After the devices verify their provisioning outputs, an authenticated assignment activates a task and selects committees according to the task-time policy described in Section I. Abstractly, the assignment is

$$\mathsf{asg} = (\mathsf{bid}_k, \mathsf{task}, \mathsf{range}, \mathcal{S}, \sigma_{\mathsf{asg}}),$$

where range identifies unused tuple-row indices within the exact batch bid_k, and $\mathcal{S} = (S_1, \ldots, S_m)$ is a partition of $[n]$. The signature σ_{asg} authenticates all preceding fields, or an equivalent authenticated control-plane mechanism provides the same binding. For row q, define the globally scoped tuple identifier $\mathsf{tid}_{k,q} = (\mathsf{bid}_k, q)$. A conforming device checks the batch identifier, task identifier, tuple range, and its unique committee assignment before delegation, and atomically marks each delegated $\mathsf{tid}_{k,q}$ as consumed in a ledger keyed by bid_k.

Let $\kappa(i) = j$ when $i \in S_j$. Device D_i delegates its MAC-key share α_i once for the activated batch/task in a message bound to $(\mathsf{bid}_k, \mathsf{task})$. For every selected tuple row q, it also sends the following five logical component records:

$$\{(\xi_i, \gamma_i(\xi), \mathsf{bid}_k, \mathsf{task}, \mathsf{tid}_{k,q}) : \xi \in \{a, b, c, r_x, r_y\}\}$$

to $C_{\kappa(i)}$ over a confidential and authenticated delegation channel. Each logical record is bound to the same batch, task, and tuple identifier. The protocol does not require five separate network transmissions; an implementation may serialize the five records in one authenticated delegation message. The receiving committee verifies that bid_k and task match the assignment and that $\mathsf{tid}_{k,q}$ names a row in its assigned range before computing

$$\bar{\alpha}_j = \sum_{i \in S_j} \alpha_i, \qquad \bar{\xi}_j = \sum_{i \in S_j} \xi_i,$$
$$\bar{\gamma}_j(\xi) = \sum_{i \in S_j} \gamma_i(\xi) \pmod p.$$

Because the S_j form a partition,

$$\sum_{j=1}^{m} \bar{a}_j = a, \quad \sum_{j=1}^{m} \bar{b}_j = b, \quad \sum_{j=1}^{m} \bar{c}_j = c = ab.$$

Thus, the aggregate rows remain valid authenticated multiplication-triple shares.

Masked Input Sharing: Input ownership is independent of preprocessing ownership. Let O_x and O_y be the authorized owners of the online secrets. For $u \in \{x, y\}$, committee C_j holds $\bar{r}_{u,j}$, $\bar{\gamma}_j(r_u)$, and $\bar{\alpha}_j$. The committees send their aggregate mask shares to O_u or an authorized input gateway, which reconstructs

$$r_u = \sum_{j=1}^{m} \bar{r}_{u,j}$$

and broadcasts

$$\epsilon_u = u - r_u.$$

The authenticated task assignment designates a public committee index $j_u^\star$ that absorbs this public correction. The committees set

$$\bar{u}_j = \begin{cases} \bar{r}_{u,j} + \epsilon_u, & j = j_u^\star, \\ \bar{r}_{u,j}, & j \neq j_u^\star, \end{cases} \qquad \bar{\gamma}_j(u) = \bar{\gamma}_j(r_u) + \bar{\alpha}_j \epsilon_u.$$

Then

$$\sum_{j=1}^{m} \bar{\gamma}_j(u) = \gamma(r_u) + \alpha \epsilon_u = \alpha u.$$

Applying this procedure independently to the edge secret x and the model-side secret y yields $\langle x \rangle$ and $\langle y \rangle$. Phase 2 completes only after every preprocessing share required by the selected tuple range has been delegated and every required application input has been injected successfully. The edge recipients and input owners may then leave the sustained MPC interaction unless the application requires them to receive or authorize the final output.

H. Committee-Level Online Multiplication Phase

At the start of Phase 3, the m committees already hold authenticated shares of the two inputs and of a fresh multiplication triple. The TEE is absent and the original edge recipients no longer participate in the interactive arithmetic. Committee C_j forms

$$\bar{d}_j = \bar{x}_j - \bar{a}_j, \quad \bar{\gamma}_j(d) = \bar{\gamma}_j(x) - \bar{\gamma}_j(a),$$
$$\bar{e}_j = \bar{y}_j - \bar{b}_j, \quad \bar{\gamma}_j(e) = \bar{\gamma}_j(y) - \bar{\gamma}_j(b).$$

After opening $d = \sum_j \bar{d}_j = x - a$ and $e = \sum_j \bar{e}_j = y - b$, each committee computes the value share

$$\bar{z}_j = \bar{c}_j + d\bar{b}_j + e\bar{a}_j + \mathbf{1}_{j=j^\star} de,$$

where $j^\star$ is the public committee designated to absorb the correction de. The corresponding output-MAC share is

$$\bar{\gamma}_j(z) = \bar{\gamma}_j(c) + d\bar{\gamma}_j(b) + e\bar{\gamma}_j(a) + \bar{\alpha}_j de.$$

These shares reconstruct $z = xy$ and preserve its global MAC authentication.

For the set $\mathcal{V}$ of opened d, e, and z values in a batch, each committee computes

$$\Delta_j = \sum_{v \in \mathcal{V}} \rho_v \big(\bar{\gamma}_j(v) - \bar{\alpha}_j v\big),$$
$$\mathsf{accept} \iff \sum_{j=1}^{m} \Delta_j = 0 \pmod{p},$$

where the nonzero public coefficients ρ_v are derived from a common challenge after the checked values are fixed [17]. A distributed realization must commit and open the committee contributions so that a corrupted committee cannot choose its residual after observing honest residuals. The protocol aborts on an inconsistent sharing or opening.

IV. Security Analysis

This section separates two guarantees. First, the simulation-based claim covers TEE-assisted generation and TEE-to-device distribution and is conditional on the enclave behaving as the attested functionality; it is therefore different from the trust guarantee of MASCOT or LowGear. Second, the implemented online path is justified by the algebraic preservation of authenticated sharing under committee aggregation and by the final MACCheck. The provisioning theorem does not by itself prove the security of an application-specific assignment service or delegation network; those channels must satisfy the authentication, confidentiality, and single-use requirements stated below.

A. TEE Trust Assumptions

We model the TEE as an ideal functionality $\mathcal{F}_{\mathsf{TEE}}$ within an otherwise untrusted server. It provides the following guarantees:

- **Isolation:** The internal state of $\mathcal{F}_{\mathsf{TEE}}$ remains opaque, and the predefined code executes without unauthorized interference.
- **Measured Code Integrity:** Execution is bound to MRENCLAVE; changing the loaded code or initial configuration changes the measurement.
- **Remote Attestation:** Quote generation and verification bind the reported measurement and report data to genuine SGX attestation evidence.
- **Volatile and Sealed State:** Per-invocation signing and session keys remain volatile, while the epoch seed and membership-scoped hash-chain state are sealed to the enclave identity. SGX sealing protects their confidentiality and integrity but does not prevent a malicious host from restoring an older valid state. Because such rollback can enable replay or reuse of preprocessing material, a production deployment must maintain the latest accepted epoch and counter in an external append-only log, quorum checkpoint, or equivalent non-rollbackable state service. Without this mechanism, our security claims are restricted to executions in which the sealed state and recipient checkpoints are not rolled back or forked.

B. Adversary Capabilities and Network Model

During provisioning, the adversary statically corrupts a subset I_D of the n recipients and may expose their keys and shares or deviate arbitrarily. For one activated task, it may also corrupt a subset I_C of the m committees; a corrupted committee learns all shares delegated to it. Committee-level privacy requires the existence of at least one uncorrupted committee whose aggregate contains at least one original device share that has not been exposed to the adversary. In particular, an uncorrupted committee is not sufficient if all original device shares contributing to its aggregate are already known through I_D. A fresh epoch or secure refresh is required before changing a partition that reuses MAC material.

The adversary controls the untrusted host and both communication networks and may eavesdrop, inject, suppress, delay, reorder, or replay messages. The implemented receiver protects the TEE-to-device path, while the online executor detects malformed partitions, invalid preprocessing relations, incorrect products, and inconsistent MACs. A multi-host realization additionally requires authenticated assignment and confidential, authenticated, replay-protected delegation from the surrounding application. The base provisioning path requires all n device shares for every activated tuple. If any source device fails before delegating its required share and no recovery mechanism is enabled, the committees cannot form a complete committee-level sharing; the affected tuple range is therefore unusable and the corresponding task execution aborts.

C. Formal Security of Authenticated Preprocessing

We use the real–ideal paradigm for a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ that statically corrupts a proper subset of the recipients.

Ideal Authenticated-Preprocessing Functionality. For an epoch eid, ordered set $\mathcal{D}$, and field $\mathbb{F}_p$, $\mathcal{F}_{\mathsf{AP}}$ samples a nonzero MAC key α and its device shares. For each authorized batch uniquely identified by $\mathsf{bid}_k = H(\texttt{'beaver-bid-v1'} \parallel h_{\mathsf{man}} \parallel \mathsf{ctr}_k \parallel \mathsf{com}_{\mathsf{epoch}})$, it samples l independent tuples $(a^t, b^t, c^t, r_x^t, r_y^t)$ with $c^t = a^t b^t$, authenticates every component, and privately delivers only D_i's local batch-labeled output. Batches in one epoch reuse the MAC-key shares but not the tuples; changing the ordered recipient set requires a new epoch. Public leakage is limited to identifiers, parameters, encoded lengths, and suppressed delivery.

Definition 1 (Security): Protocol Π realizes $\mathcal{F}_{\mathsf{AP}}$ with abort if, for every PPT $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for every PPT environment $\mathcal{Z}$,

$$|\Pr[\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda) = 1] - \Pr[\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{AP}},\mathcal{S},\mathcal{Z}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

Proof Sketch: The simulator obtains corrupted recipients' prescribed outputs from $\mathcal{F}_{\mathsf{AP}}$ and invokes $\mathcal{F}_{\mathsf{TEE}}$ for attestation. Under decisional Diffie–Hellman (DDH) and HKDF security, honest-recipient session keys may be replaced by uniform keys; AEAD security then permits encryption of equal-length zeros, while ECDSA unforgeability protects the manifest and transcripts. These hybrids are indistinguishable, and the adversary can only suppress delivery. Hence Π realizes $\mathcal{F}_{\mathsf{AP}}$ with abort for static recipient corruption. Definition 1 and this simulation argument cover only authenticated preprocessing generation and TEE-to-device distribution. They do not cover task-time committee assignment, device-to-committee delegation, share aggregation, input injection, or online multiplication.

D. Conditional Security of Committee-Level Execution

The post-provisioning path is supported separately from Definition 1. By linearity, the aggregation map in Section III preserves the global sums of the value shares, MAC shares, and MAC-key shares and therefore preserves both the Beaver relation and the global MAC invariant; the online Beaver computation and final MACCheck then retain their standard authenticated-sharing guarantees [3], [17], [18]. This argument requires a complete one-time partition, a valid authenticated assignment, confidential, authenticated, and replay-protected delegation, and at least one uncorrupted committee whose aggregate contains at least one original device share that has not been exposed to the adversary. These are explicit system assumptions, not consequences of the preprocessing simulation proof.

E. Malicious Behavior Analysis

Identity and Context Binding

The deployment authority signs the ordered recipient list and batch parameters, and the enclave verifies this manifest before using any pk_i. The manifest hash is included in the Quote context and every recipient transcript, while the Quote binds the enclave signing key to $(h_{\mathsf{man}}, \mathsf{ctr}_k, \mathsf{com}_{\mathsf{epoch}})$, whose domain-separated hash is bid_k. Consequently, a hostile host cannot substitute a recipient key, enclave key, or batch context without forging a signature or causing receiver rejection.

The signed KEYENC record binds the ephemeral ECDH point, salt, counter nonce, manifest, and recipient index. AES-GCM tags protect individual frames, whereas the exact row count and final transcript signature detect deletion, duplication, reordering, and cross-recipient substitution. Because recipient ECDH keys are static, compromise of sk_i can expose previously recorded sessions. The encrypted payloads contain consumable preprocessing triples rather than private application inputs; accordingly, the current protocol does not provide forward secrecy.

Committee Assignment Integrity and Single Use

A conforming task assignment binds $(\mathsf{bid}_k, \mathsf{task}, \mathsf{range}, \mathcal{S})$ to one verified provisioning batch. Each device verifies that it appears in exactly one S_j, accepts only tuple identifiers of the form $\mathsf{tid}_{k,q} = (\mathsf{bid}_k, q)$ within the assigned range, delegates each such identifier once, and rejects conflicting assignments. Every delegation message and consumption-ledger entry carries the same bid_k; committees reject a mismatch among the assignment, delegation payload, and tuple identifier. The implemented executor checks $1 < m < n$ and that the selected

sets form a complete, duplicate-free partition before aggregation. These checks prevent cross-batch mixing as well as local omission or duplication from entering an online execution. Issuance of signed assignments and a globally fork-resistant, batch-scoped consumption ledger remain responsibilities of a distributed deployment.

Optional Failure-Triggered Public Share Reassignment

The base additive provisioning is n-of-n: every original source share required by an activated tuple must be available before committee aggregation. For large deployments, a simple optional recovery policy can trade a bounded amount of privacy margin for fault tolerance. Let F be the set of devices declared failed before delegation. If $|F| \leq f_{\max}$, an authenticated failure-control mechanism may release, from a recovery copy prepared during the offline provisioning stage, the complete authenticated local share of each $D_i \in F$, including α_i, the value shares, and the corresponding MAC shares. A designated stable holder absorbs the released share component-wise; for example,

$$a'_j = a_j + \sum_{i \in F_j} a_i, \qquad \alpha'_j = \alpha_j + \sum_{i \in F_j} \alpha_i,$$

with the same addition applied to b, c, r_x, r_y and all MAC shares. Linearity preserves the represented values, Beaver relation, global MAC key, and MAC invariant, so subsequent committee aggregation and online multiplication are unchanged.

Public release intentionally exposes the failed devices' additive shares. Before committee delegation, preprocessing confidentiality therefore requires

$$|F \cup I_D| < n,$$

so that at least one original device share remains unknown to the adversary. Each released share consumes one unit of this privacy margin, and failure declarations must be authenticated to prevent an adversary from falsely triggering disclosure. If $|F| > f_{\max}$, or if the remaining privacy condition would be violated, the current epoch is terminated rather than releasing additional shares: the deployment authority issues a new recipient manifest, a fresh MAC epoch is created, and the TEE performs a new offline provisioning run for the surviving or replacement recipients. This recovery policy is a deployment-level extension that preserves the TEE's offline-only role because any releasable recovery copy must be prepared during offline provisioning.

Data Confidentiality and Integrity

Under AES-GCM authenticity, the probability that a modified ciphertext frame is accepted satisfies

$$\Pr[\mathsf{Dec}_{K_i}(\widetilde{C}_i) = \mathsf{accept}] \leq 2^{-|\mathsf{tag}|} + \mathsf{negl}(\lambda).$$

Combined with transcript verification, this protects the confidentiality, integrity, ordering, and completeness of an honest recipient's provisioning output.

Replay Detection and Rollback Limitation

Under the stateful-recipient assumption, a recipient accepts only the expected membership-scoped counter transition and batch identifier relative to its uncompromised local checkpoint. Consequently, it rejects an old transcript and any mixture of artifacts from distinct generation attempts under one batch identifier.

This guarantee does not provide global rollback or fork resistance. A malicious host can restore an older sealed state or target a new recipient that lacks the latest checkpoint, potentially causing preprocessing material to be replayed or reused. As required by the TEE trust assumptions above, production deployment therefore needs an external append-only log, quorum checkpoint, or another non-rollbackable state service.

Resistance to Recipient-Key Leakage

Before delegation, compromise of any proper subset of device keys reveals only the corresponding additive shares; a complete triple and the global MAC key remain hidden while at least one original device share is unknown. After delegation, the effective corruption unit changes to the committee aggregates. Committee-level privacy requires at least one uncorrupted committee whose aggregate contains at least one original device share that has not been exposed to the adversary. Mobile corruption that accumulates aggregate MAC-key shares across different partitions of one epoch remains outside the claim.

V. Performance Evaluation

A. Prototype Implementation

Because the offline stage is the primary performance bottleneck, it is the focus of our evaluation. Our scheme implements manifest validation, sealed epoch derivation, buffered SGX randomness, authenticated preprocessing over the 127-bit prime field $\mathbb{F}_{2^{127}-1}$, session establishment, AES-GCM streaming, and Quote binding. Field elements are stored internally in the 128-bit C++ type `unsigned __int128`. The untrusted host handles DCAP infrastructure and persistent ciphertext, while a standalone receiver verifies the complete transcript before committing plaintext. To bound transient enclave memory use during streaming, each streaming work buffer is capped at 64 KiB; this per-buffer limit is distinct from the configured `StackMaxSize` and `HeapMaxSize`. A task-time executor consumes only receiver-verified plaintext, validates a complete device partition, aggregates each group's value, MAC, and MAC-key shares, performs masked input recovery and Beaver multiplication, and executes a batched MACCheck.

B. Complexity Analysis

For n recipients and l triples, enclave generation, sharing, hashing, and encryption require $O(nl)$ field or symmetric operations; per-recipient ECDH and signatures add $O(n)$ public-key operations, and Quote generation is $O(1)$ per batch. Each device performs one ECDH exchange, verifies one Quote and

TABLE I
PROTOTYPE AND EXPERIMENTAL CONFIGURATION

Item	Configuration
Processor	Two-socket Intel Xeon Silver 4316 system at 2.30 GHz; 20 physical cores per socket (40 physical cores and 80 logical CPUs in total)
Execution parallelism	One software thread (`NLWP=1`)
Memory	16 × 32 GB DDR4 modules (512 GB total), operating at 2666 MT/s
Operating system	Ubuntu 22.04
Evaluated dimensions	Offline recipient count n and triple count l
Network setting	Single-server loopback and local committee execution

TABLE II
MEASURED COMMUNICATION AND STORAGE BREAKDOWN FOR $n = 5$, $m = 2$, AND $l = 8$

Component	Bytes	Scope
Signed manifest	560	Per batch
KEYENC negotiation	363	Per device
Quote and public metadata	10,488	Per device
Encrypted payload	3,198.8	Per-device mean
Binary-frame overhead	32	Per frame
Complete recipient file	14,113.8	Per-device mean

three application-layer signatures—the manifest, KEYENC, and transcript signatures—and decrypts $O(l)$ data. Task-time delegation transfers $O(nl)$ field elements and aggregation performs $O(nl)$ additions. For each injected private input, masked input sharing requires the m committees to deliver one aggregate mask share to the input owner (or authorized input gateway) and one public correction ϵ_u to be broadcast back. Thereafter each Beaver product requires local work linear in m and openings among only the $m < n$ logical parties; a batched MACCheck also processes the opened values and their m MAC shares. These task-time and online operations are implemented and functionally evaluated.

Under fixed-width encoding, each recipient's ten field elements per row contribute $160l$ logical bytes. Key establishment adds 168 raw bytes. In the evaluated binary wire format, each AES-GCM frame adds a 12-byte initialization vector (IV), a 16-byte tag, and a 4-byte length prefix, for 32 bytes of framing overhead. Each recipient file also carries batch metadata, a transcript signature, and a hexadecimal copy of the Quote. The signed manifest has $240 + 64n$ bytes.

Table II reports a fresh release-mode hardware-SGX run in the integrated configuration used below. It separates per-batch, per-device, and per-frame costs; entries marked as means are averaged over the five recipient files.

C. Experimental Comparative Analysis

Experimental Environment

Note: In practical deployments, `StackMaxSize` and `HeapMaxSize` should be carefully tuned to the recipient count and workload. Over-provisioning these sizes increases Enclave initialization and teardown time due to Enclave Page Cache (EPC) allocation (adding 0.1s to 4.5s in our experiments, scaling with size). Furthermore, on EPC-constrained systems, excessive allocation can induce frequent EPC paging, severely degrading overall throughput. Unless otherwise stated, the reported offline generation time includes enclave creation and teardown, triple/MAC generation, per-recipient packaging, authenticated encryption and hashing, protocol computation, and the full DCAP attestation path, as well as outside-call (OCALL) transition/copy overheads. All experiments use the single-server setting in Table I and exclude SFTP/WAN transfer.

Experimental Data

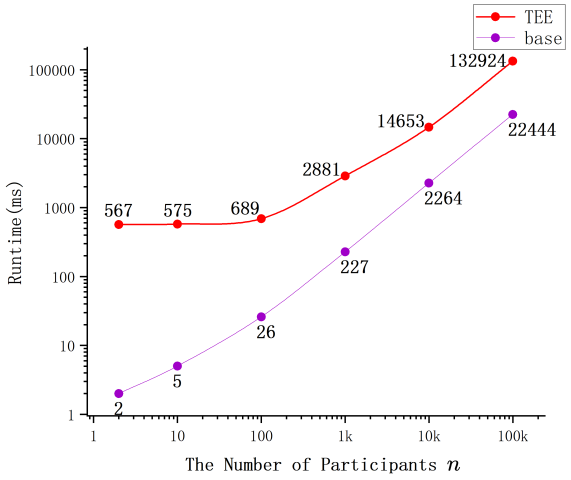


Fig. 2. Impact of Edge-Device Recipient Scale on Provisioning Time

Fig. 2 compares the complete TEE execution with the no-TEE native baseline at $l = 1$ while varying the number of recipients, thereby separating the fixed cost of the protected provisioning path from recipient-dependent work. For small recipient sets, enclave creation and DCAP dominate: the TEE runtime is 567–575 ms for $n = 2$–10, whereas the native baseline requires only 2–4 ms. As n increases, per-recipient key establishment, sharing, and packaging become dominant and both curves approach linear growth. At $n = 100{,}000$, the TEE provisions one row for every recipient in 132,924 ms, or approximately 752 recipient files per second, compared with 22,444 ms for the native baseline. The resulting $5.92\times$ TEE-to-native ratio is substantially smaller than at low n, showing that the fixed SGX/DCAP cost is amortized, although protected per-recipient processing remains measurable. This stress test isolates local provisioning and excludes independent client decryption, assignment, delegation, and online execution.

Fig. 3 evaluates generation time along the orthogonal triple-count dimension in the three-recipient configuration and shows the same amortization effect. Up to $l = 1$k, the complete TEE path remains between 565 and 593 ms, while the native baseline rises from 3 to 21 ms, so fixed enclave and attestation costs dominate. The 10M point corresponds to approximately 34.84k globally shared triples per second, and the streaming design bounds enclave memory pressure. Thus, the security and attestation overhead is most significant for small jobs and is progressively amortized by large batches. This experiment excludes storage and online consumption. The 100k-recipient and 10M-triple stress points evaluate $(n, l) = (100{,}000, 1)$ and

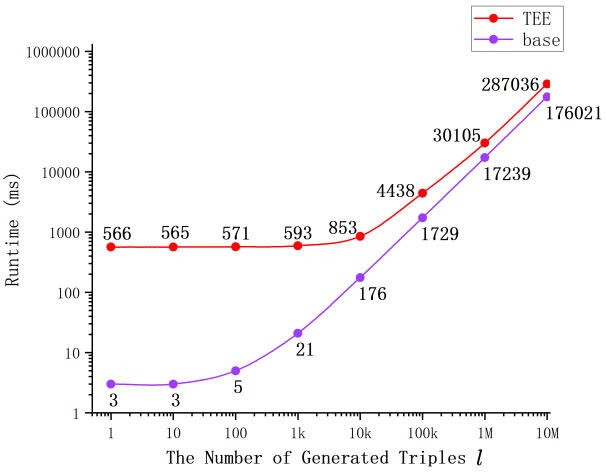


Fig. 3. Impact of Triple Production Scale on Time Consumption

$(3, 10^7)$ separately and do not claim simultaneous operation at both maxima.

To further evaluate the scalability limit, we tested the implementation with 65,000 participants and 65,000 authenticated triples on real SGX hardware. The experiment completed successfully in 55,608 seconds with a 256 MiB enclave heap. This configuration processed approximately 4.225 billion participant–triple output units and produced a logical ciphertext volume of 1.708 TB. Under the same 256 MiB enclave-heap limit, however, the implementation crashed when configured with 80,000 participants and 80,000 authenticated triples. Such a large configuration is supported by the streaming generation and distribution architecture, in which authenticated shares are incrementally encrypted, hashed, and flushed instead of retaining the entire $O(nl)$ output in enclave memory. Consequently, memory consumption is primarily determined by the participant state and bounded upload buffers, while computation time and output volume grow with $O(nl)$. This result establishes 65,000 participants and 65,000 triples as the largest successfully completed configuration in our experiments, demonstrating a high practical scalability limit despite the single-threaded implementation.

Integrated Offline-to-Online Validation

To validate the scheme as one continuous data path rather than as disconnected components, we executed a fresh batch with $n = 5$, $m = 2$, and $l = 8$. Release-mode hardware SGX generated the authenticated triples and masks in binary mode, and all five independent receiver invocations verified the three application-layer signatures (manifest, KEYENC, and transcript), DCAP Quote, ECDH/HKDF key establishment, AES-GCM frames, field ranges, and exact row count before releasing plaintext. The verified device outputs were then partitioned into committees of sizes three and two and passed directly to the online executor. For each row, the executor aggregated the device shares, recovered authenticated shares of synthetic edge- and model-side inputs from independent masks, executed Beaver multiplication, verified the output, and accepted the batched α-MAC check.

The same run included two negative tests. Flipping one ciphertext byte caused receiver rejection at the AEAD path, while adding one field unit to an online output-MAC share caused the final committee MACCheck to reject. This experiment confirms that genuine enclave output is consumed by the implemented committee protocol without a synthetic-share boundary. Consequently, the result establishes functional protocol integration and tamper detection.

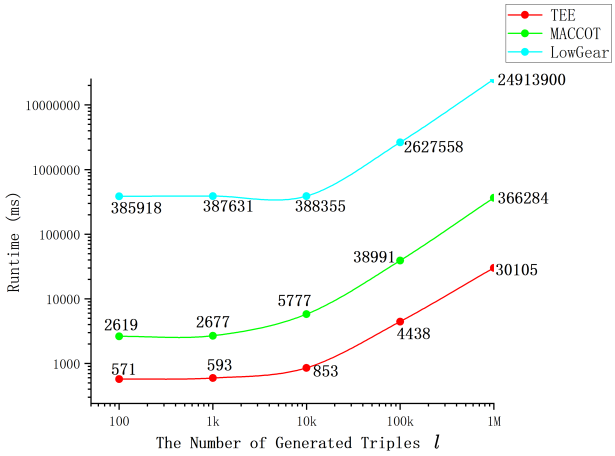


Fig. 4. Performance Comparison

Fig. 4 compares the TEE service with the evaluated MASCOT and LowGear configurations in MP-SPDZ [19]. Within the evaluated single-server implementations, the TEE service is the fastest at every tested triple count, and its relative advantage increases as larger batches amortize its fixed enclave and attestation costs. MASCOT grows steadily with the triple count, whereas LowGear incurs substantial HE setup and computation costs. At $n = 1000$ and $l = 1$, the evaluated MP-SPDZ implementations terminated because their single-host process, communication, and system-resource overhead exceeded the available resource budget. Since these costs increase with the number of parties, configurations with $n > 1000$ were deemed infeasible under the same hardware and operating-system limits and were therefore not executed.

These runtime results represent a trust–performance trade-off rather than a direct comparison under identical security assumptions. The proposed service relies on SGX confidentiality and integrity, DCAP attestation, and the deployment authority, whereas MASCOT and LowGear avoid a trusted dealer but rely on OT- or HE-based computational assumptions and malicious-security checks. Accordingly, the reported speedups characterize only the evaluated implementations, single-server environment, and respective trust models; they are not protocol-intrinsic speedups under equivalent security assumptions. The Appendix provides the detailed functional and trust comparison.

Taken together, Figs. 2–4 identify two performance regimes. Small jobs are dominated by fixed enclave-lifecycle and DCAP costs, whereas large batches amortize these costs and increas-

ingly favor the proposed TEE service over the evaluated native and software-only baselines. Recipient scaling is also approximately linear at large n. These measurements support the intended use of the design as a high-throughput offline service for large preprocessing batches and device populations, rather than as an optimization for very small batches. The integrated execution complements the scaling results by confirming that the generated material remains consumable by the committee protocol and that ciphertext and online-MAC tampering are rejected.

VI. Conclusion and Future Work

This paper presents an attested TEE-assisted MPC scheme that confines the trusted enclave to the offline generation and per-device distribution of authenticated multiplication triples and input-recovery masks. At task time, selected device shares are delegated to and aggregated by late-bound committees, which perform masked input recovery, Beaver multiplication, and MACCheck without re-entering the TEE. Under the explicit assumption of an uncompromised Intel SGX enclave, the prototype provides remote attestation and encrypted delivery and carries verified preprocessing material through the integrated committee computation. This device-first design trades $O(nl)$ offline storage and delegation volume for a flexible online compute topology. In separate stress tests, the prototype supports $(n, l) = (100{,}000, 1)$ and $(3, 10^7)$; it was not evaluated with 100,000 recipients and 10 million triples simultaneously. These single-server results demonstrate scalability along two independent dimensions under the evaluated implementation and trust model, while the integrated tests confirm offline-to-online usability and tamper rejection.

Several directions merit further investigation. First, production deployments should mitigate microarchitectural leakage through constant-time execution and, where necessary, oblivious random-access machine (ORAM) techniques, while evaluating the resulting performance cost. Second, extending the framework to heterogeneous TEE platforms, such as Arm TrustZone and Hygon Confidential Secure Virtualization (CSV), would reduce dependence on a single vendor and enable comparisons of their attestation, sealing, and resource constraints. Third, supporting matrix triples and edaBits would extend the preprocessing pipeline to a broader range of MPC workloads, with future evaluation focusing on the associated storage, communication, and computation tradeoffs.

References

[1] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge Intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[2] H. Zhang, W. Fan, and J. Wang, "Bidirectional utilization of blockchain and privacy computing: Issues, progress, and challenges," *Journal of Network and Computer Applications*, vol. 222, p. 103795, 2024.

[3] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[4] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, 1991, pp. 420–432.

[5] M. Keller, E. Orsini, and P. Scholl, "MASCOT: Faster malicious arithmetic secure computation with oblivious transfer," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 830–842.

[6] M. Keller, V. Pastro, and D. Rotaru, "Overdrive: Making SPDZ great again," Cryptology ePrint Archive, Paper 2017/1230, 2017. [Online]. Available: https://eprint.iacr.org/2017/1230

[7] E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl, "Efficient pseudorandom correlation generators: Silent OT extension and more," Cryptology ePrint Archive, Paper 2019/448, 2019. [Online]. Available: https://eprint.iacr.org/2019/448

[8] P. Miao, A. Murphy, A. Srinivasan, and M. Tromanhauser, "Pseudorandom correlation generators for multiparty Beaver triples over $\mathbb{F}_2$," Cryptology ePrint Archive, Paper 2025/1182, 2025. [Online]. Available: https://eprint.iacr.org/2025/1182

[9] V. Rieder, "Silentium: Implementation of a pseudorandom correlation generator for Beaver triples," in *International Conference on Applied Cryptography and Network Security*, 2025, pp. 296–317.

[10] I. Gamiz, C. Regueiro, O. Lage, E. Jacob, and J. Astorga, "Challenges and future research directions in secure multi-party computation for resource-constrained devices and large-scale computations," *International Journal of Information Security*, vol. 24, no. 1, p. 27, 2025.

[11] M. Hoekstra, R. Lal, P. Pappachan, C. V. Rozas, V. Phegade, and J. Del Cuvillo, "Using innovative instructions to create trustworthy software solutions," in *Proc. 2nd Int. Workshop Hardware and Architectural Support for Security and Privacy*, 2013, article 11.

[12] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution," in *Proc. 2nd Int. Workshop Hardware and Architectural Support for Security and Privacy*, 2013, article 10.

[13] Intel Corporation, "Remote attestation for multi-package platforms using SGX DCAP," Intel, Online technical documentation, 2025. [Online]. Available: https://cc-enabling.trustedservices.intel.com/intel-dcap-mp-ra/01/introduction/

[14] P. Scholl, N. P. Smart, and T. Wood, "When it's all just too much: Outsourcing MPC-preprocessing," in *Cryptography and Coding*, 2017, pp. 77–99.

[15] R. Rachuri and P. Scholl, "Le mans: Dynamic and fluid MPC for dishonest majority," in *Advances in Cryptology – CRYPTO 2022*, 2022, pp. 719–749.

[16] S. D. Gordon, D. Starin, and A. Yerukhimovich, "The more the merrier: Reducing the cost of large scale MPC," in *Advances in Cryptology – EUROCRYPT 2021*, 2021, pp. 694–723.

[17] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Annual Cryptology Conference*, 2012, pp. 643–662.

[18] M. Keller, P. Scholl, and N. P. Smart, "An architecture for practical actively secure MPC with dishonest majority," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, 2013, pp. 549–560.

[19] M. Keller, "MP-SPDZ: A versatile framework for multi-party computation," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1575–1590.



Puyu Wang (Student Member, IEEE) received the B.S. degree in information security from Xidian University in 2023, where he is currently pursuing the Ph.D. degree in cyberspace security. His research interest is privacy-preserving computation, including multiparty computation and trusted execution environments.



Ruidan Su (Member, IEEE) is a professor with the School of Cyber Engineering, Xidian University, China. He received the Ph.D. degree in computer science and technology from Xidian University, China, in 2010. He has published over 20 papers in journals and conferences. His research interests include network and information security, cryptographic applications, data security, and privacy protection.



Zhenshen Liu (Student Member, IEEE) received the M.S. degree from the School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin, China, in 2024. He is currently pursuing the Ph.D. degree at Xidian University, Xi'an, China. His research interests include federated learning, differential privacy, and network resource allocation in the IoT.



Ruiqi Yang (Student Member, IEEE) received the B.S. degree in information security from Xidian University in 2025, where he is currently pursuing the Ph.D. degree in cyberspace security. His research interest is post-quantum cryptography algorithms, including post-quantum secure channels and authenticated key exchange.



Kai Fan (Member, IEEE) received the BS, MS and PhDdegrees from Xidian University, China, in 2002, 2005 and 2007, respectively, in telecommunication engineering, cryptography and telecommunication and information system. He is working as a professor with the State Key Laboratory of Integrated Service Networks at Xidian University. He published more than 70 papers in journals and conferences. He re ceived 9 Chinese patents. He has managed 5 national research projects. His research interests include IoT security and information security.



Hui Li (Member, IEEE) was born in Shaanxi, China, in 1968. He received the B.S. degree in radio electronics from Fudan University in 1990, and the M.S. and Ph.D. degrees in telecommunications and information systems from Xidian University, in 1993 and 1998, respectively. He is currently a Professor with Xidian University. His research interests include network and information security.

APPENDIX
FUNCTIONAL COMPARISON AND SGX/DCAP PROVISIONING WORKFLOW

Fig. 5 shows the prototype's generic SGX/DCAP workflow for context. Table III compares the evaluated methods' trust assumptions and measured runtimes relative to base.

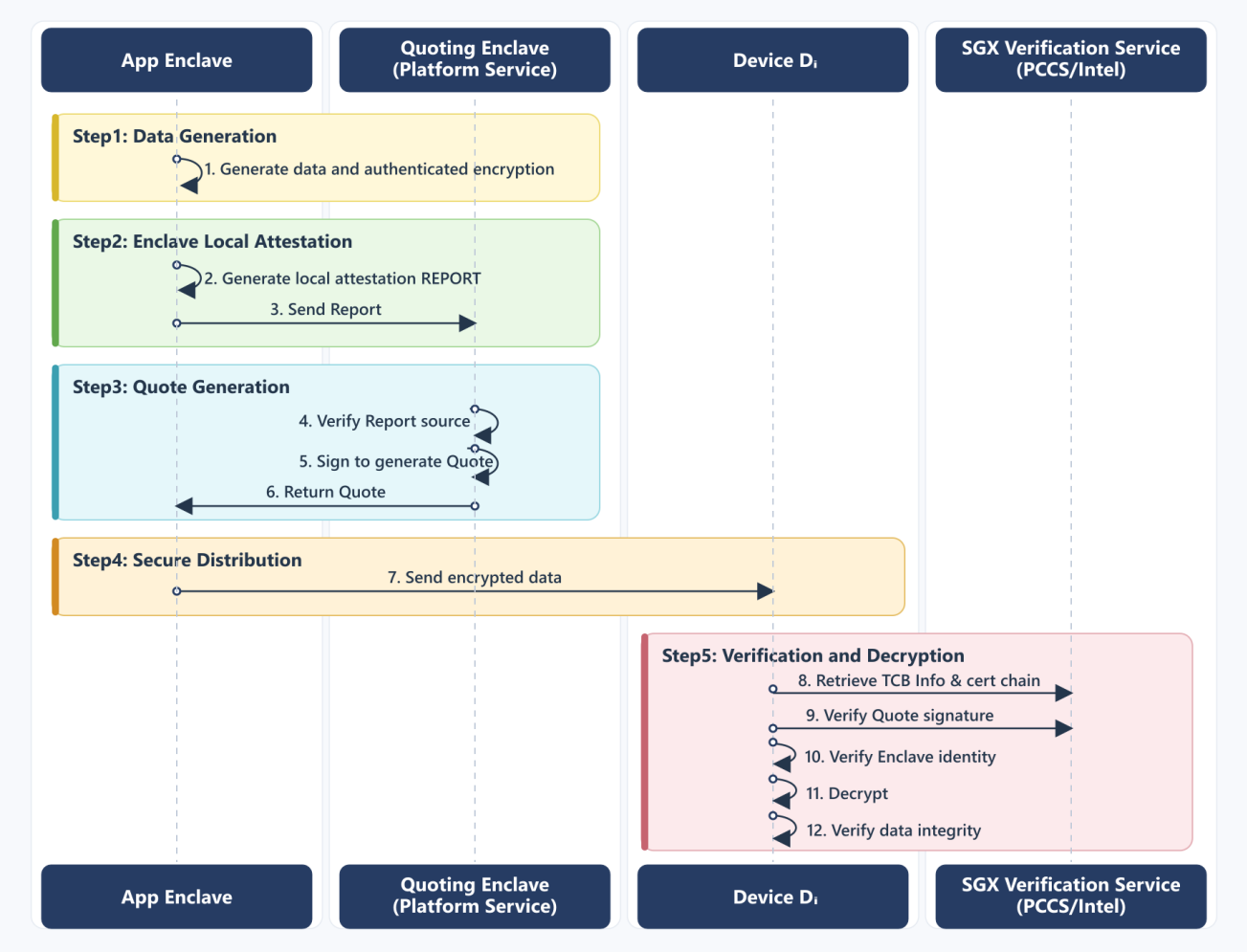


Fig. 5. Generic SGX/DCAP provisioning and distribution workflow used by the prototype.

TABLE III
FUNCTIONAL, TRUST, AND RELATIVE-RUNTIME COMPARISON

Method	Trust assumption	Recipient work	Offline interaction	Attestation	Relative runtime (base $= 1$)
Proposed TEE	SGX confidentiality/integrity, deployment authority, and DCAP	ECDH/HKDF, Quote and signature verification, and $O(l)$ AEAD decryption	One encrypted delivery per device; task-time delegation	DCAP	$6.47\times$ at $(10^4, 1)$; $2.57\times$ at $(3, 10^5)$
MASCOT [5]	No trusted dealer; OT-based computational assumptions and malicious-security checks	OT extension, authenticated correlation generation, and sacrifice	Interactive multiparty preprocessing	No	– at $(10^4, 1)$; $22.55\times$ at $(3, 10^5)$
LowGear [6]	No trusted dealer; homomorphic-encryption assumptions and malicious-security checks	HE operations and distributed consistency checks	Interactive multiparty preprocessing	No	– at $(10^4, 1)$; $1519.70\times$ at $(3, 10^5)$
Base	Generator is trusted with triples and MAC material	Receive, verify, and store shares	One-way share delivery	No	$1.00\times$ at both workloads

Workloads are $(n, l) = (10^4, 1)$ and $(3, 10^5)$; "–" denotes no measurement. MP-SPDZ exhausted resources at $n = 10^3, l = 1$, so $n > 10^3$ was infeasible under the same single-host resource limits.